Fingerprint-based indoor positioning and intensity classification using an improved machine learning framework

by

Kushant Patel

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Sciences
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2021

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Covid-19 has changed the world in terms of business, public and many other fields. Millions of livelihoods have been affected by the pandemic. Amidst the upheaval, work from home and restrictions on indoor gatherings have played a significant role in flattening the curve. Even after enforcing restrictions, numbers are still on the rise. Various covid-19 tracing applications have been designed to keep track of positive cases. There is an increased need of tracking positive covid cases to limit the spread of the virus to ordinary people. The continents are trying to flatten the curve and maintain a good economic condition to attain normalcy in the season of chaos. Technology has proved helpful in times of pandemics. Now we have IoT devices and advanced tech, including cameras, Wi-Fi, Bluetooth, RFID etc., which can be used for tracking positive patients. This tracking should be made efficient without exploiting the privacy of users. Vaccination research along with proper tracking seems to be a failsafe solution for evading covid-19 after effects. Amidst all these available strategies, Indoor localization seems to be one of the required fields of research. This thesis dives into establishing a machine learning framework that can be used across all kinds of IoT (Internet of Things) systems and WSNs (Wireless sensor networks). Distance estimation based on fingerprint has been a widely researched field for indoor localization algorithms. Several traditional approaches have been tried out , including trilateration, triangulation which needs more testing parameters and renders them complex. Fingerprinting techniques seems to be helpful. Even though various fingerprinting techniques have been tried out, we do not have a generic framework that can be used for research on fingerprinting. The system has been implemented as a part of cloud remote monitoring solutions and an accurate blend of ensemble bagging and boosting methods for making an accurate distance estimation based on the strength of RSSI fingerprints. The framework hopes to serve as a base platform for all kinds of indoor localization research. It encompasses a BLE based system that acquires data from leak detection systems, relays it to the cloud via BLE gateway, accumulates data on a cloud database and is passed as alert notifications to users via the use of cloud designed app. At the other end, the database aids in the creation of a location dataset for machine learning which is used for training the model. A regression machine learning model is deployed for the prediction of distances based on fingerprint strength which can be utilized for various fingerprint algorithms. A classification machine learning model is deployed for fingerprint intensity classification to evaluate fingerprint levels in different environments.

Acknowledgements

I would like to acknowledge all those who have been supporting me throughout my research work. The completion of this master's journey would not have been possible without their help. My sincere gratitude to my supervisors Prof. George Shaker and Prof. Norman Zhou for their valuable guidance and support throughout my research work. I appreciate their professional supervision, strong support, encouragement, and patience over the past two years. I appreciate reading committee for taking time out to review thesis. Special thanks to Prof. James Tung for taking time out for evaluation of the thesis.

My sincere thanks go to wireless sensor devices and the lab's team – Nathan, Nimesh, Martins, Connor and all other team members.

Finally, I would like to thank my family for their unconditional love and support. Special thanks to my father, Dhirubhai, my mom, Shushilaben, my brother, Nishant and my sister-in-law, Shreya, for supporting me financially and emotionally in my life. I would like to thank Team Watonomous for pushing me into new directions of research. Last but not least, this research wouldn't have been possible without endless love, full support and warm encouragement of all my dearest friends who motivated me to pursue the field of robotics.

Dedication

To

My Family & My dearest friends

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Indoor localization involves the retrieval of a device or a user's location in all kinds of indoor environments. It is almost a decade since the introduction of smartphones, IoT devices etc. Tracking and localization enabled in these domains have has led to the design of various applications to improve route navigation and advertisements. These applications range from museums gallery navigation to police and firefighting systems. Innovative architectures have been developed to aid in healthcare, building management and surveillance sectors. Nowadays, various heterogeneous data sources have made information available, easy and reliable. Big data and machine learning has become the common buzzword in software industry. Emerging technologies like Wi-Fi, ZigBee, UWB, Bluetooth, Bluetooth Smart are indivisible parts of these IoT networks. Localization is classified into three major categories - Device-based localization (DBL), Monitor based localization and Proximity detection. Device-based localization relies on reference or anchor nodes to narrow down a location. This kind of localization can be used for navigation, mostly in the case of structures like museums, art galleries, tech expo etc. The second category, i.e., monitor-based localization, is like the first one in the way that it uses reference nodes. However, this one is used for tracking assets and provision of services based on tracking.

Proximity categorization of the localization is a reliable and affordable solution using estimation of the distance between the user and the area of interest. Traditional approaches have been effective in the estimation of distances for short-range based on reference points. Satellite-based approaches are meant for outdoors and fail to provide an accurate estimation of distances in indoor environments. Contact tracing applications have become essential for the prevention of the spread of Covid-19 (*Coronavirus Disease (COVID-19) Situation Reports*,2019). Efforts need to be made in this arena because manual testing takes time and puts the people performing this testing at risk. Bluetooth Low Energy or BLE signals are readily available on most of smartphones which can be used for estimation of position as smartphones are almost everywhere. But again, there is a significant limitation of this approach – not all people have smartphones with them every time. This reduces the capacity of tracing positive cases via the use of BLE tracing applications. However, smartphones, laptops and other smart devices are connected to the internet all the time. So, we can use Wi-Fi or BLE or a mixed approach that can estimate position efficiently, thereby improving the indoor localization approach. But before we reach there, we

need a common base platform for research. The approach discussed in this thesis is aimed at the design of this framework which can be further used for all kinds of indoor localization research.

1.2 Research Objectives

Determining the location and knowledge of the user surroundings have become essential factors for navigation and the design of intelligent indoor structures. There is no doubt that GPS has come out on top for navigation and localization. It has become a vital technology in our daily lives. Since its introduction in 1973 by the US Department of Defence, GPS gives the position of any object globally relative to earth. Satellite deployment has become affordable and reliable with the exponential progress in the design of rockets. GPS lets us know our current position as well as our past locations. In contrast, it's an excellent tool for military, commercial applications, driverless cars, vehicle tracking systems. The accuracy of GPS signals decreases in urban areas where signals are weak due to various EMI, RF interferences, obstacles etc. Emerging technologies, including Bluetooth, RF, Ultrasound, Wi-Fi etc., have given rise to IoT connectivity. The concept of the interconnectivity of people, devices and technologies has transformed the world of localization.

Deterministic approaches like trilateration and triangulation have the ability to localize user/device with accurate distance estimations, but they need complex equipment like the angle of arrival measurements, base stations making the measurement process cumbersome. Region-based estimations use a relatively more straightforward mechanism via the use of signal strength maps or fingerprint databases with spatial context for accurate distance estimations. Offline data is used for training machine learning models which learn and tries to predict the distance based on supervised learning algorithms. The online phase uses trained models for real-time prediction. With the pandemic season at its prime phase, we need better frameworks for testing positive cases as well as ensure safety amidst the vaccination phase. Indoor localization approaches based on machine learning algorithms have achieved accurate distance estimations. Approaches like Triangulation and Trilateration have played an important role in location determination and asset tracking. These approaches need complex setup and the equipment cost is high which makes them difficult to use in several cases where fingerprinting solution could prove useful. Fingerprinting solutions do not need complex equipment, but they suffer from issue of multipath fading. Machine learning combined with fingerprinting solution is a useful approach to evade multipath fading. There is a need for a base platform or a generic machine learning framework for indoor localization research. A reliable approach that ingests data from sources incorporates spatial context, and a machine learning pipeline might lead us to a new perspective of seeing the problem of indoor localization. Deployment of such a framework can be further used for contact tracing apps without exploiting the

privacy of users. The aim of the thesis work is to reduce the complexity factor and provide a cheap and open-source platform for indoor localization.

1.3 Contributions

I. Dataset generation – Major part of any machine learning use case involves dataset generation and model generation. This framework has been used for creation of a new location dataset. Reference datasets have been used in the past, but when it comes to the domain of indoor localization, datasets might vary from place to place due to interferences involved. The framework introduced in the thesis solves these issues:

   a. Reduces efforts in dataset generation for any kind of fingerprints used for indoor localization.

   b. The use of any standard reference dataset is also supported; in fact, one of the major datasets has been used for the evaluation of the framework.

II. Novel Water Leak Detection Systems for acquisition of RSSI Fingerprints (Witham *et al.*, 2019) – A new data ingestion pipeline via use of cloud approach has been designed for acquisition of sensor data including RSSI fingerprints and used for creation of a location dataset with labelled outputs generated using dataset generation algorithm. This dataset can be used for further analysis of water leakages for indoors.

III. Model generation – The machine learning pipeline introduced covers the use of supervised learning algorithms. The algorithms can vary from domain and environmental conditions. For example, some algorithms, including decision trees, K-nearest neighbours and cat-boost, have been covered.

IV. Regression plus Classification – The regression phase can be used for distance estimation using a generalized distance metric (Minkowski's distance – Euclidean and Manhattan distance). The classification phase has been introduced to cover fingerprint intensity classification. This intensity classification has been introduced based on some standard scales. The fingerprint used in the thesis is RSSI (Received Signal Strength Indicator), but other fingerprints can be used with the proposed framework.

V. Low-code framework – With the use of a low code framework, any research candidate with minimal prior experience in python or machine learning can easily use it. The code for both dataset generation and model generation will be open-sourced for use.

1.4 Thesis Organization

The overall structure of thesis is composed of six chapters. The first chapter introduces the proposal of the framework with introduction of problem statement, research objectives and major contributions. The second chapter covers the Bluetooth architectures and indoor localization using triangulation, trilateration, and fingerprinting. The chapter mentions advantages and disadvantages of approaches used. The third chapter comprises of literature survey conducted on indoor localization via use of machine learning algorithms. A brief description on classification and regression is done for evaluation using several fingerprints and supervised learning algorithms. The fourth chapter starts with cloud implementation for data acquisition and moves onto dataset generation and model generation. A reference dataset along with transformed dataset are run through dataset generation and then through model generation. The fifth chapter presents the evaluations of the model generated. Different plots are used for evaluation of classification and regression phase. The final chapter concludes the thesis with a future scope.

Chapter 2
Bluetooth and Indoor Localization

2.1 Introduction

This chapter summarises the studies on Bluetooth and indoor localization. The architecture of Bluetooth and Bluetooth Low Energy (BLE) has been discussed in detail with a focus on profiles, services and characteristics. Wireless sensor networks (WSNs) have become crucial for our day to day lives. Emerging as an active research area, WSNs have found their practical roots in energy consumption, robustness, localization, sensor locations, routing algorithms and so on. Composed of a finite set of sensor devices distributed indoors and outdoors environment, these networks aim is to acquire valuable environmental insights and relay the information further for data analysis which can drive forecasting and alert systems. Low power, low data rate and reliable communication nodes make up WSNs Bluetooth and Wi-Fi are two of the major communication protocols which can be utilized for WSNs. This chapter throws a light on these major protocols with a focused analysis on BLE, which consumes less power. A review of some of the existing indoor localization approaches has been done to draft out a comparison on limitations and advantages of traditional as well as fingerprinting approaches.

2.2 Bluetooth

Bluetooth has been one of the significant breakthroughs of the 20th century. This widely used short-range wireless communication protocol was first introduced in the 1990s as an alternative to RS-232 cable. The protocol in its abstracted form is being used prominently in mobile phones, portable computers, sensors, headsets etc. It was developed as an IEEE standard (IEEE 802.15.1) which is now maintained by a Special Interest Group or SIG. Bluetooth is now available in six versions, the latest one being Bluetooth 5.0 is commonly used in wireless hardware, audio hardware as well as game controllers, keyboards, mice. This version is revolutionizing the existing state of the art of IoT with almost twice the speed, four times the range and eight-time the broadcasting of the message capacity (Bloem and Schiphorst, 2011) . Cost efficiency, low power wireless and easy deployment have become the holy grail of IoT. Following represents the original architecture of Bluetooth -

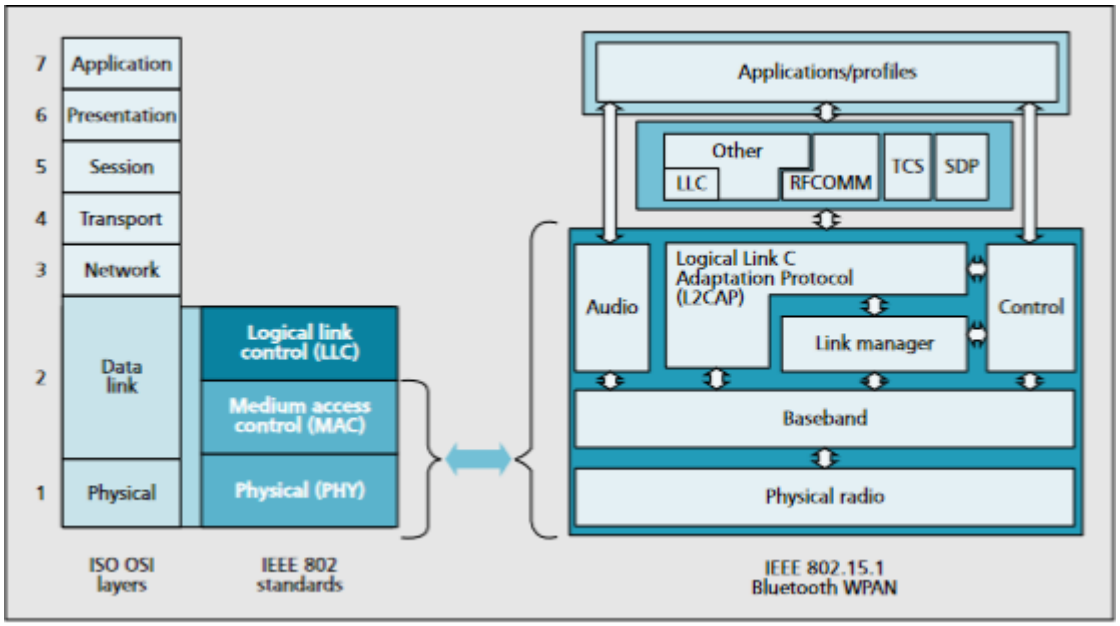


Figure 2-1. Bluetooth architecture IEEE 802.15.1

The architecture delineates a radio interface and a reliable communication stack. The interface and stack allow all devices to search for each other and advertise the services. The devices use the 2.4 GHz ISM band for Bluetooth communication. The physical radio layer modulates/demodulates data for transmission/reception over 2.4 GHz band radio frequencies and illustrates the transceiver's physical characteristics. Moving up, we have baseband layers that define the framing, timing, packets and flow control on the link. The link manager layer is responsible for authentication, security, QoS, transmission scheduling and power consumption. The Control layer provides a coherent interface to hardware manufactured by various other manufacturers. This is ensured via a command interface to baseband levels and link managers. The L2CAP or Logic Link Control Adaptation Protocol handles the connection-less services to upper layers, including protocol multiplexing, Segmentation, QoS support and reassembly of the protocol data units from upper levels. RFCOMM provisions emulation over a serial port which facilitates existing applications operating on serial communications. HCI (Host Controller Interface) which is not visible in the current architecture diagram, presents an interface between the device driver and the software aspect of the system. SDP (Service Discovery Protocol) is crucial for provisioning the interface to the link controller, thereby granting interoperability between various Bluetooth devices.

2.3 Bluetooth Low Energy

A low power wireless technology is operating in the 2.4 GHz ISM band, targeting applications with low power and the ability to run on batteries for years. This technology was introduced in Bluetooth 4.0 in 2010 (Afaneh,2016). BLE has found its profound use in sensor data, low bandwidth applications and controlling of devices. The connections established using BLE are quicker than classic Bluetooth. The major version update is Bluetooth 5 (Bluetooth Smart), which the prominent version over all previous BLE standards. This version offers data rates in four different transmission ranges - 125 kbps, 500 kbps, 1Mbps and 2 Mbps. BLE uses a GFSK modulation scheme hence allowing max data throughput of 1Mbps. BLE makes cell operated applications like fitness wearables quite feasible. In January 2020, Bluetooth SIG introduced version 5.2, which brought a significant change in LE Audio. ISOC, LEPC and EATT are three significant features of Bluetooth 5.2. It is backwards compatible, which means you can continue using devices using Bluetooth 4.2 with Bluetooth 5.0. Bluetooth has two main advantages – low cost and low power consumption, which makes it ideal for any kind of low power device for daily use. It uses almost a fifth of the power of Wi-Fi. When it comes to positioning APIs, Bluetooth becomes an alternative to Wi-Fi API for indoor positioning. The IoT market has been revolutionized because of Bluetooth. With Wi-Fi having too many parameters, the process to localize devices for indoor positioning becomes complex. Time synchronization is one crucial feature missing from Bluetooth, which renders time-based triangulation methods difficult in terms of implementation. The second major issue is the occasional use of directional antennas being a low footprint device without which angle measurement becomes challenging. BLE or Bluetooth Low energy (Gomez, Oller Bosch and Paradells, 2012) – a power-friendly version introduced for devices that needed to be run off a tiny cell for long periods. BLE became an essential standard for low power sensors, which would be used as wearable tech. Smartwatches, heartbeat monitors, tracking ids, etc., are some of the commonly used devices which make use of BLE. This version supports central operating systems including apple OS, windows 8, GNU/Linux, Android 4.3+ etc.

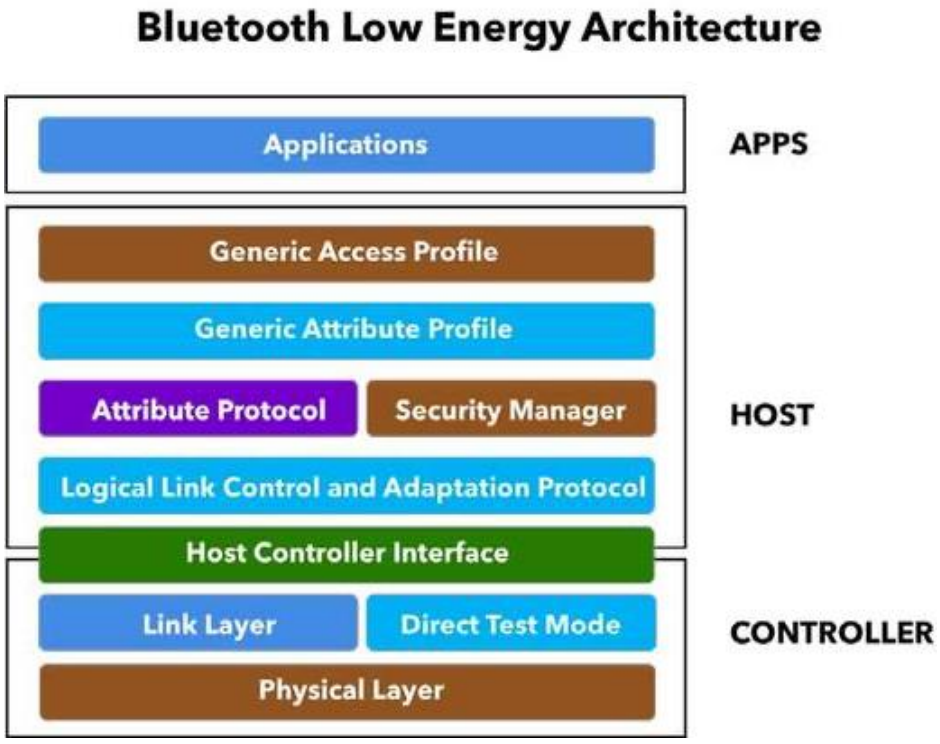


Figure 2-2. Bluetooth Low Energy Architecture

The architecture of BLE follows pretty much like Bluetooth architecture, including the physical layer, link layer, L2CAP layer. The physical layer is here again used for modulation and demodulation of data over radio interface operating in frequencies specified by 2.4 GHz ISM band. Link-layer abstracts higher levels via the use of HCI. Timing requirements and management of radio state are also handled by this layer. HCI acts as a standard protocol allowing the host layer for communication with the controller layer. L2CAP layer is used for protocol multiplexing accepting various protocols from upper layers and relaying them to lower layers. There is the inclusion of a Direct Test Mode, which tests the radio operation in terms of transmission power, receiver sensitivity etc.

2.3.1 GAP (Generic Access Profile)

GAP makes the device accessible to the outside world. It handles the communication between two devices and controls connections and advertising. Two types of roles are defined by GAP - Peripherals and Central devices. Peripheral devices are low powered, resources constrained devices like a BLE tag, smartwatch etc. Central devices are medium powered devices like a phone or a gateway with better processing power and more memory.

2.3.2 GATT(Generic Attribute Profile)

GATT makes use of an Attribute Protocol for storing services, characteristics and related data via the use of a LUT(Lookup Table) with 16-bit identifiers. GATT implementation usually comes after GAP; once the two devices have successfully established communication, GATT will use services and characteristics to define this communication for transferring data back and forth between two devices. Only one peripheral device is allowed to connect with the one central device following exclusivity. The other devices cannot connect once the peripheral device is connected to the central device. The service/Client relationship is the major part of GATT. The GATT server or the peripheral device holds the lookup data and definitions for services and characteristics. GATT client sends a request and receives a response from this server. A connection interval is specified by the peripheral to a central device, during which the central device will keep reconnecting to verify any new data availability (Townsend, 2020).

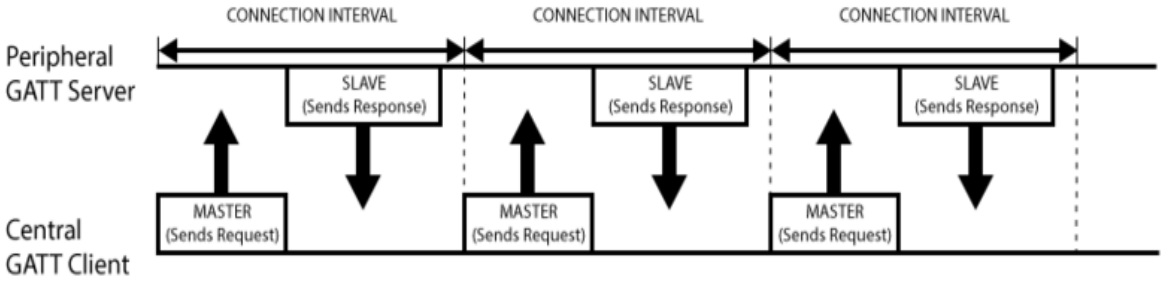


Figure 2-3. GATT implementation

2.3.3 Profiles, Characteristics and Services

Profiles are pre-defined collections of designer compiled services. GNSS profile is one of the profiles prepared by Car Working Group, which specifies how a GNSS client, including a laptop and phone, can run a navigation stack and obtain positioning data from the server without the presence of a GPS receiver (*National Marine Electronics Association - NMEA*, no date). Characteristics are used for encapsulation of a single datapoint and have predefined 16 bit or 128 bit UUID. These present the lowest level of GATT transactions that interacts with the BLE peripheral. Services contain one or more characteristics and identify themselves from each other via the use of 128-bit identifiers.

2.4 Wireless Fidelity (Wi-Fi)

Wi-Fi is a prevalent wireless networking technology operating in the RF bands IEEE 802.11b, IEEE802.11 n, IEEE802.11 g in 2.5 GHz and IEEE802.11 a in 5Ghz range (Crow *et al.*, 1997). Wi-Fi is

widely distributed in all indoor environments making it the most reliable means of indoor localization. Almost all devices, including cameras, tablets, cell phones, smart devices and laptops, use Wi-Fi for communication and for operation. It's cheap, affordable and widely available. Wi-Fi localization uses RSSI based fingerprinting approach. The current Wi-Fi system supports a data rate of 54 Mbps with an indoor coverage of 100 feet. IEEE 802.11 family operates over a larger 20 MHz bandwidth. Below is the technical comparison of three major Wi-Fi standards.

Feature	**802.11a/g**	**802.11 b**
Channel Bandwidth	20 MHz	25 MHz
Duplexity	Half-duplex	Half-duplex
Modulation	BPSK, QPSK, 16/64 QAM	QPSK
Application	Wireless LAN	Wireless LAN
Bandwidth	2.7 bps/Hz	0.44 bps/Hz
Access protocol	CSMA/CA	CSMA/CA
Encryption	RC4m	RC4

Table 2-1. Comparison of major Wi-Fi standards.

IEEE 802.11 uses CSMA/CA or Carrier Sense Multiple Access / Collision Avoidance and are half-duplex where transmission and reception are on the same channels. 802.11 has included a distributed control function to facilitate transmission only in clear channels to avoid collision detection. Security has been a significant concern on wireless technologies. 802.11 family has an RC4 based 40/104-bit encryption with a static key. Wi-Fi operates on radio signals with transmitters responsible for transmitting it over a modulated channel with receiver antennas receiving it. Access points (APs) are a significant source of transmission and reception of radio waves. Wi-Fi cards are included in devices; these can be internal or external; a USB antenna attachment is also available. A PCMCIA card is included in the laptop, which connects to access points for data. Hotspots are created by access point installation and connection to the internet. Hotspots are available in cafes, hotels, book stores, university areas, etc., to allow

accessibility to the internet. ISPs (Internet Service Providers) are the major suppliers of Wi-Fi networks(Bloem and Schiphorst, 2011).

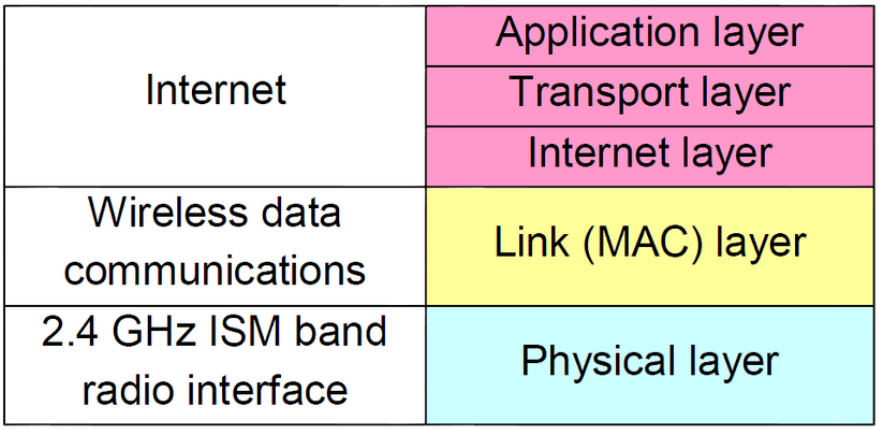


Figure 2-4. Wi-Fi generic architecture

The architecture is composed of the physical layer, MAC layer, Application/Transport/Internet layer. Physical layers perform encoding/decoding of signals, includes the transmission medium specification, handles bit transmission/reception. MAC layer assembles data frame and error detection and address at the transmission end and disassembles data frame with error detection and address recognition at the reception end. The link-layer provides an interface to higher layers and is responsible for flow and error control.

2.5 Receiver Signal Strength

Receiver Signal Strength Indication or RSSI – a radio signal strength is used to infer the mobile node's location. Three or more points in three-dimensional space are used to predict the exact location of the node. RSSI is used for determining the optimal radio energy of a link and measured in Decibels or dB. Received signal strength is a very economical factor for Location-Based Services due to its vast presence in almost all wireless devices. The core stage of RSSI based positioning relies on reference radio maps with point measurements. RSSI based positioning is done in two stages - an online and an offline stage. The offline stage mainly deals with data collection, whereas the online stage performs online estimation. A database is it using data collected in the offline stage, including spatial aspects of the indoor environment. A radio map can be collected using some known algorithms and stored online/offline in a relational database. The radio-map built consists of (x, y and z) location coordinates of each point of an

indoor structure. The approach of RSSI based positioning makes several assumptions. Some of them are mentioned below -

- Reference points - Visibility is enough even with variations to identify them uniquely.
- Transmitters and receivers - transmission is assumed to be the constant signal strength, and radio receiver characteristics are similar for both locating and mapping devices.
- Environment - a significant contribution is a factor of the environment that would not change when the radio map is built.

RSSI fingerprinting is further described in detail in the fingerprinting section.

2.5.1 RSSI Ranging

The relationship between transmitted power and received power of wireless signals and distance between the nodes is given by the following equation (Adewumi, Djouani and Kurien, 2013) –

$$\mathbf{Pr} = \mathbf{Pt} \cdot \frac{\mathbf{Gr\,Gt}}{(\mathbf{4\,\pi\,d}/\lambda)^2} \quad \text{…….. (Eq. 1-1)}$$

Taking logarithms and multiplying by 10 on both sides on eq. (1-1)

$$\mathbf{10\,lg\,Pr} = \mathbf{10\,lg\,Pt} - \mathbf{10\,n\,lg\,d} \text{….. (Eq. 1-2)}$$

Where;

Pt refers to the transmitted power of the wireless signals.

Pr refers to the received power of the wireless signals.

Gt and Gr refers to antenna gain of transmitter and receiver.

λ refers to the wavelength of the signal.

d denotes the distance between the sending nodes and reception node.

n refers to the transmission factor, and its value depends on the propagation environment (Arthi and Lochana, 2019).

After conversion, eq. (2) describes the power in dBm or decibel-milliwatts.

$$\mathbf{Pr(dBm)} = \mathbf{A} - \mathbf{10n\,lg\,d} \text{ …… (Eq. 1-3)}$$

2.6 Indoor Localization

Positioning and localization have always been significant areas of research. A big change was brought with the rise in Global Navigation Satellite Systems, which led to the birth of American based Global Positioning Systems and Russian GLONASS in the 1980s and 1990s. GNSS is primarily concerned with position accuracy with millimetres and centimetres positions and access to these accurate locations to users with relevant apparatus. Initially born with military objectives, GNSS became commercially available with the advent of smartphones. The increased popularity of smartphones led to application development making use of GNSS services. As more people started using location-based services, including google APIs for navigation, context-aware recommendation systems, etc., accuracy became a crucial factor in the localization problem. A reliable means of real-time localization was obtained using GNSS applications, but localization accuracy drops inside indoor structures due to various factors despite its success on outdoor localization. GNSS systems are affected by signal blockage, multipath propagation issues, Non - line of sight problems. This decreased faith of GNSS to be used indoors.

As homo sapiens spent more of their time indoors, the use of GNSS systems for indoor localization was discouraged. Indoor localization is the problem of dealing with location, position and navigation. Location is a higher-level abstract of position, for e.g. location of some restaurant in some mall. The position usually refers to standard coordinates in the form of latitude and longitude. Navigation defines the logic of traversing from point A to point B. The major limitation of using current technologies lies in inability to estimate count of people indoors. The problem is attributed to noise induced in form of interference in indoor positioning and localization. Various algorithms have been implemented for proximity detection. Location-based services refer to applications that make use of location APIs for services like navigation, tracking, marketing, data acquisition etc. The problem with Global Positioning System (GPS) is that power of the signal goes on decreasing with the distance, which makes it difficult to be used for indoor localization. Wi-Fi access points are an expensive way of indoor localization, so; we try to perform a strategic search through BLE based indoor localization algorithms. The environment model includes several beacons/mesh nodes and an agent. The test space is treated as a flat space with background interferences from floors, signals, walls etc. Indoor positioning algorithms are classified into two major types – one with the need for prior measurements and another without the need for prior measurements. The former classification includes algorithms that use proximity, centroid, weighted-centroid, trilateration, and the latter involves fingerprinting-based methods.

2.6.1 Localization techniques

1. Angle of Arrival

The direction of the radiofrequency wave is determined by the angle of arrival when propagation of wave encounters several antennas, which is measured by time difference when the wave propagates through each antenna. This is commonly known as The Time Difference of Arrival and is used for the estimation of the location of wireless devices.

2. Time of Arrival

Figure. 2-5 denotes two signals arriving at the antennas, where some delay exists in the time arrival of both the signals at the antennas, another measurement method. The distance can be easily calculated based on the speed of these signals, given their time of arrival. It is difficult to measure this time unless some form of synchronization is done between the transmitters and receiver's clock. This tricky part of time calculation increases complexity in its implementation.

3. Distance-based

Using a path loss model of RSS, estimation of position is done. This approach needs a mean accuracy of 4 meters and emitters position (Bose and Foh, 2007).

4. Proximity-based

According to GSM, smartphones are assigned a cell for connection and accuracy is determined by distance calculation between the cells.

5. Inertial based

Inertial sensors can be used for data acquisition. The data contains information relating to the orientation and speed of sensors. This data proves to be useful for localization. Lot of calibration and resetting integrator is critical to the implementation of inertial based localization systems.

VI. Fingerprinting

A widely used approach without the need of base stations and complicated equipment for estimation of position. A reference radio map is built using RSS, which relates values with the positions.

2.7 Indoor localization approaches

The estimation methods make use of distance estimation to various Access Points (Aps). The estimation methods which use signal characteristics to determine location are referred to as fingerprinting methods.

- Triangulation

The name suggests that this approach somehow relies on the triangle's geometrical calculations. The triangulation approach considers two receivers, with one source emitting a signal towards them(Hartley and Sturm, 1997). The angle of arrival is calculated for each signal arriving at the receiver as depicted in the following figure-

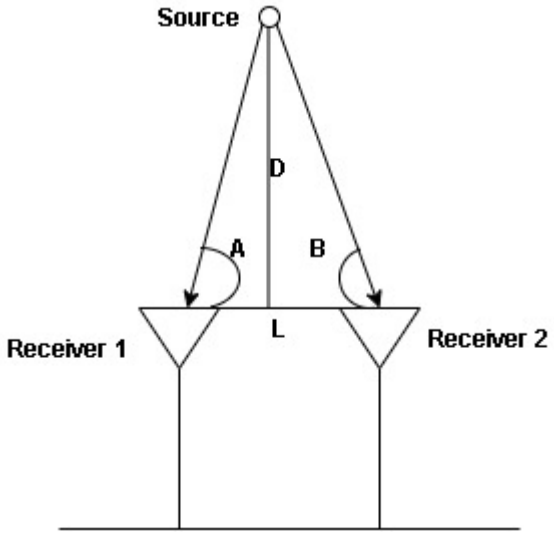


Figure 2-5. Experimental setup for Triangulation

Angles A and B denote the Angles of Arrivals for Receiver 1 a, and two respectively, L denotes the length between the receiver and D can be formulated from the following equation –

$$\mathbf{D} = \mathbf{L}\frac{\mathbf{Sin\ A + Sin\ B}}{\mathbf{Sin(A+B)}} \ldots\ldots\ldots \text{(Eq. 1-4)}$$

- Trilateration

One of the prominently used methods for position estimation, commonly used in navigation and GPS. For these, three or more Access points or Aps are needed. RSSI calculation is done between the device and Access points for considering the radius from AP on which device might be located. The estimation of device or source positioning is done by the intersection of three radii of spheres resulting from Aps. It is also referred to as the Line-of-sight measurement method; the obstacles might affect the RSSI and time of propagation(Sturgess and Carey, 1987). This Line-of-sight method is sensitive to disturbances, especially at large distances. Consider an assumption of noise introduction in the model around 3dBm, which results in the ranging uncertainty of the same order of magnitude as the source distance. As the distance increases, this method could become unreliable for measurement. Floorplan construction based on this method is unsuitable.

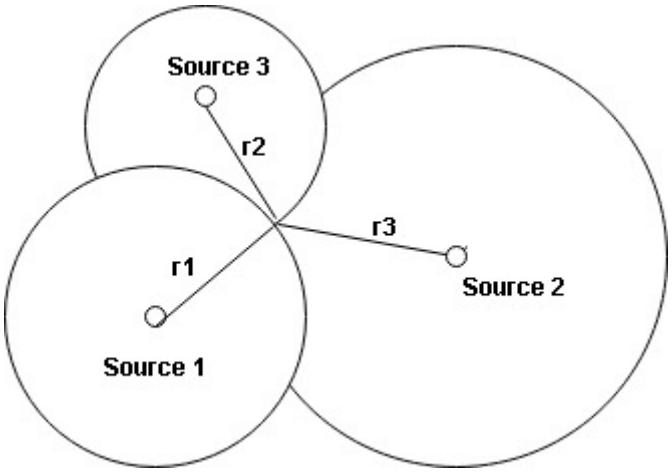


Figure 2-6. Estimation of position using Trilateration

- Odometry

  Estimation of the motion is possible using odometry and, this is all possible with smartphone sensors consisting of the accelerometer for detection of forwarding motion. The magnetometer gives the current heading; over time, odometry cannot be relied on as it is susceptible to drift. However, when combined with the fingerprinting methods, odometry can provide an absolute estimate of the position. The challenge with this method is – it makes estimations based on position relative to the last known position, so unless we know at least one position, we cannot proceed ahead. A drift from integrators also poses a major challenge.

2.8 Fingerprinting

Fingerprinting is used for estimation and is the most popular available method for accurate estimation. It runs on an identification of location based on specific sensor measurements and signal properties. This approach needs no additional infrastructure to be deployed as Wi-Fi fingerprints are available almost everywhere. These signal properties and their location are stored in some central repository or database, which can be used for mapping radio properties at different locations. The significant advantage of this method over any approaches discussed so far is not relying on the Access point locations. It can work even without determining the exact locations of these Access points. Fingerprinting using Wi-Fi prints is able to achieve 2-3 meters of accuracy. The indoor propagation is affected by multipath fading induced by electromagnetic and radio-frequency interference. The implementation can be improved with the use of advanced algorithms which can identify hidden patterns. (Subedi and Pyun, 2017). Generalized

patterns can then be used for improving accuracy even in the presence of significant interferences. Machine learning algorithms are one way of looking at this problem for the identification of valuable patterns. Artificial Neural Networks(ANNs), Bayesian classifiers, KNNs, decision trees are some of the standard algorithms which are being used in indoor localization. Fingerprinting approaches follow two phases, the offline phase being a step where the signal properties are collected, and radio mapping is done, whereas the online phase deals with the matching of data acquired with the radio map and determining the location. A brief look into both the phases has been described below –

1. Offline Phase

   The phase starts with the setting up of an environment essential for fingerprint-based positioning. The phase consists of acquiring data, radio mapping and creation of the central repository or database to hold everything. The test area is divided into subareas, and we provide reference points for each sub-area. Graph-based methods can be used for this. A unique identification needs to be added for each of the reference points, which depicts Access points and their RSSI. Hence, each reference point gets a unique identification with signals passing via several walls, devices, obstacles, humans etc. These unique identifications are referred to as fingerprints. As RSSI varies over time due to obstacles, humans etc., multiple samples need to be taken to avoid missing out essential fingerprints, which can then be embedded as a list or vector and stored in the central repository/database.

2. Online Phase

   This phase is much simpler than the offline phase. It deals with data acquisition from the device in the form of RSSI from the Access Points around it. The list/vectors with identifications (ids) and names are sent to the server with RSSI. On the server-side, matching of fingerprints is done to provide accurate locations. The localization algorithms search for the best fingerprint match using a fingerprint database. The estimation of distance can be formulated by minimization of Euclidean distance -

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{xj}} \sum(\mathbf{ai} - \mathbf{xj})^2 \text{.....} \quad \text{(Eq. 1-5)}$$

   Where ai is the fingerprint

   xj is the fingerprint from the fingerprint database

   RSSI based fingerprint approach is based on a signal attenuation model, which requires low-cost equipment and low low-cost infrastructure. This has caught the attention of research community.

However, the propagation model described above is idealistic and is far from realistic implementations where signal occlusion, electromagnetic interference and personnel flow are common factors challenging localization. Instead of looking at this from signal level interference, we need to look at this problem from a feature matching problem (Zhu *et al.*, 2020).

2.8.1 Channel State Information (CSI)

Channel state information is one of the other primary fingerprints which makes use of the information contained in different subcarriers of a single channel. CSI signals define the process of signal transmission between transmitter and receiver, including signal attenuation, delay distortion and scattering. CSI is a physical layer characteristic and can easily differentiate between multi-path signals, hence making it reliable in terms of precision and stability (Caire and Shamai, 1999). The following figure describes various fingerprints used for indoor localization –

Fingerprint	**Advantage**	**Disadvantage**
RSSI	Easily available, low-cost and easy deployment.	Susceptible to multipath effect, noise interference with the training model, inaccuracies in positioning.
Bluetooth	Easy to collect signals, low power consumption and less expensive.	This fingerprint relies on RSSI signals which makes it susceptible to multipath effect and leads to inaccuracies in positioning.
Channel State Information (CSI)	Better stability, high precision and less susceptibility to multipath effect.	Random signal phase attainment due to multichannel, modification of driver needed for measurement.
Magnetic field	Natural environment signal and not affected by multipath effect	Electromagnetic interference is high in indoor environments and extra calibration cost is required.
Visible light	Easy to collect and system is scalable	Natural light interferes its path, line-of-sight is required.

Table 2-1.Advantages and Disadvantages of different fingerprints

2.8.2 Wi-Fi Fingerprinting

The approach of fingerprinting achieved using Wi-Fi involves interference from many objects including both still moving and still ones causing attenuation of signal. Diffraction , scattering and reflections due to these objects causes attenuation of signal, sometimes up to 2-3 dB even when the position of transmitter and receiver are not change (Bose and Foh, 2007) . Hence, relying on this approach becomes

unreliable as Wi-Fi radiations are absorbed by human beings and reflected by other interferences. This causes a drop in the accuracy of Wi-Fi indoor localization. The offline phase is aimed at the collection of reference points. Multiple measurements have to be considered for the offline phase as the values might fluctuate at a different time due to the

presence of humans or obstacles. A radio map is built using RSS values along with a set of all vectors acquired for all reference points. The online phase deals with the acquisition of RSS values, and position estimates are done using an algorithm. Several algorithms are available, including K-nearest neighbours, probabilistic and Bayesian algorithms. Radio maps act as a feature space for these algorithms. When the access point locations are available, any existing centroid method can be used for the estimation of position (Bose and Foh, 2007).

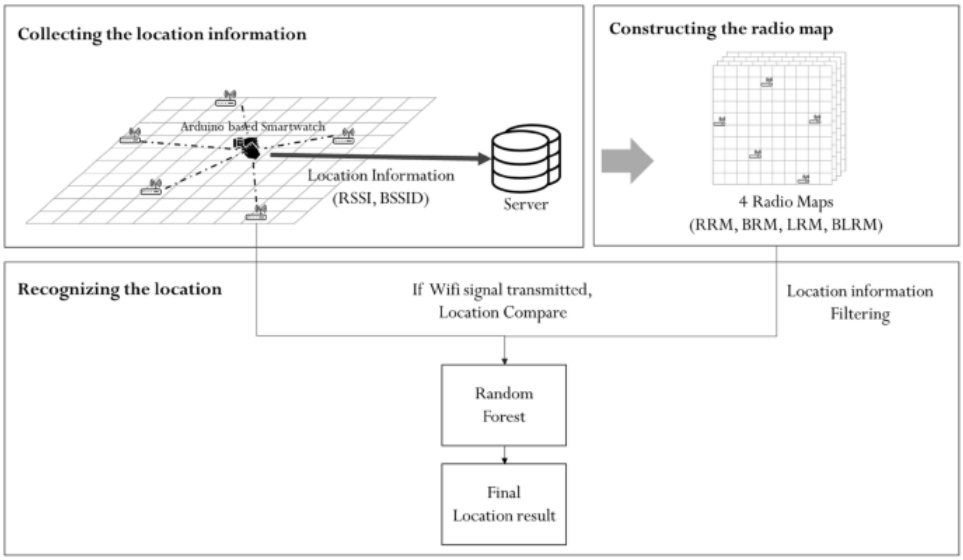


Figure 2-7. Generic Architecture of Wi-Fi fingerprinting

2.8.3 Significant Wi-Fi fingerprinting issues

As discussed before, this approach has some significant limitations. With better frameworks and approaches, we can overcome these limitations. The major issues are highlighted below-

1. Energy consumption - Applications designed for low power devices are meant for reducing energy consumption. Power consumption is a crucial factor for such devices; hence better algorithms are needed to reduce the scan intervals.

2. Interference from humans - Wi-Fi is absorbed by people, which proves to be a significant limitation as the devices on which localization applications are deployed involves many people. Better machine learning frameworks can be deployed to reduce noise or find hidden insights even amongst a vast majority of humans.
3. Variation in signal strength - Wi-Fi signal strength is varying at different locations depending on obstacles and personnel flow, causing a drop or rise in strength. With such varying strengths, estimation of position becomes tricky.
4. Radio-map creation - The creation of a Radio-map is a complicated procedure. Manual efforts are needed with accurate reference point locations in the offline phase. Human errors have to be weighed in along with an assumption that nothing is changed in setup during the offline and online phase.

2.8.4 Radio-Maps

Radio-maps are represented as a model of network characteristics which includes the spatial context. They are used for the estimation of a position using some known algorithmic approach. Wi-Fi / BLE based radio maps include the RSSI parameter along with the coordinates of the sensors/beacons/Access-points. The locations can be drafted in the form of latitude, longitude and floor locations as x, y and z coordinates. RSSI, along with the coordinates observed over different samples, forms the base radio map and serves as a fingerprint database. This database can be any local database stored offline, or it can be stored online on a remote cloud server. The distance from a central node is calculated using any known distance formula. Chapter 3 discusses the algorithmic approach for the creation of a fingerprint database in detail. The aim of the radio map is to search for the hidden pattern that governs the relationship between RSSI value and the location with spatial context added to it. The actual values obtained from sensor/beacon/access points from vendor to vendor. Environmental factors affect the mean value of RSSI observed over different time intervals. Time dependency is also weighed in when trying to estimate the positions using advanced algorithms. Data can be acquired continuously by walking at a steady pace in numerous locations. Obstacles and other interferences also need to be considered during the readings. The environmental space can be narrowed down to a rectangular grid of some specified length to observe the readings, and here, the locations specified are in terms of x , y and z coordinates where x and y are cartesian coordinates of rectangular grid and z represents the floor locations. The creation of a radio map is often a simple procedure but a time-consuming process. There are always unknown sources, including the floor size, divisions, corridors and materials of the indoor structures, which influences the creation of the radio maps. Crowdsourcing is a reliable approach to build

radio maps for large indoor structures. The main idea behind crowdsourcing is the enrolment of users on the survey and have them continuously contributing to radio-maps creation. This approach reduces the search space for large indoor structures and makes it time-consuming. Indoor maps have no specified standard which can be used as a reference. SLAM (Simultaneous Localization and Mapping) is a popular crowdsourcing approach tried out by many researchers. Graph SLAM is one such SLAM approach making use of inertial based user motion measurement and use of Wi-Fi signals sensed for building a crowdsourced fingerprint radio map (Grisetti *et al.*, 2010).

2.8.5 Challenges in the indoor positioning algorithms

1. Multipath Fading

   The major problem lies with the receiver, which fails to detect the original signal(s) from its reflections. Sometimes due to the presence of obstacles, and in other cases, the signal travels more distance than the directed path to reach the receiver, which leads to errors in the estimation of the distance, thereby raising the complexity of indoor positioning.

2. Synchronization

   Some of the methods discussed above are based on accurate synchronization between transmitter and receiver clocks, and due to multipath fading and interference from other obstacles, it is almost impossible to synchronize both the clocks.

3. Signal Propagation

   Since we rely on signal propagation and these are electromagnetic signals, attenuation over varying distances are bound to occur, following the inverse square law usually measured on a logarithmic scale (dB). The SNR (Signal to Noise ratio) worsens as the distance from the source increases and the signal becomes weaker due to all kinds of interferences.

2.9 Conclusion

This chapter summarize the use of Bluetooth and Wi-Fi architecture used for indoor positioning and localization. The use of GPS indoors raises significant issues which are highlighted. RSSI is the common fingerprint which measures the strength of a signal. This fingerprint with an improved framework is used for indoor positioning. Radio-maps are created using RSSI fingerprints. Electromagnetic interferences and obstacles (both living and non-living) cause a drop in RSSI measurements which lead to inaccuracies in positioning and therefore localization. Triangulation and trilateration are no doubt one of the best

approaches for indoor positioning but use of high-cost and complicated equipment evinces the expertise needed to carry out the process for obtaining RSSI measurements. Fingerprinting is a low cost and less reliable approach for RSSI measurements for indoor positioning purposes but use of machine learning specifically data driven algorithms make accurate measurements which outperform trilateration and triangulation approaches.

Chapter 3

Literature survey

3.1 Introduction

This chapter covers the literature survey reviews on some of the machine learning algorithms carried out during the course of research. Comparison of tried out machine learning approaches for indoor localization is discussed along with their advantages and disadvantages. Machine learning is defined as a field of study that gives computers the ability to learn without being explicitly programmed. Machine learning is a subfield of computer science where machines are meant to learn when not explicitly programmed to perform a variety of tasks. They do this by observing a pattern and try to imitate the patterns drawn from those observations to perform tasks. Machine learning is broadly classified into three main categories – Supervised Learning, Unsupervised Learning and Reinforcement Learning. Supervised learning is primarily used for the transformation of a dataset into another. This kind of learning draws a pattern based on labelled outputs. This domain of algorithms can learn patterns from one dataset and apply those patterns to new datasets and is used for making predictions. For example, prediction of stock prices or make weather predictions based on historical data.

The input dataset consists of values drawn from historical data that is used for pattern observations which are then formulated as a function that relates it to output. The formulated function is referred to as a model, which is then used for making predictions on unseen values. The model derived is the driving force behind significant breakthroughs that came in machine learning, including computer vision, Natural language processing, Autonomous vehicles etc. Supervised learning forms the foundation of applied machine learning when we have a set of input variables/features and labelled outputs, which can be used by algorithms to learn the mapping function from input to output. Unsupervised Learning is like supervised learning in a way that it can transform one dataset into another. But with unsupervised learning, there might be not a single correct answer or a single model that we are trying to derive from the dataset. Common examples of this kind of learning are clustering methods which transform datasets or datapoints observed from the dataset into a group of cluster labels. For example, in marketing firms, unsupervised learning is used for the determination of different segments of customers in terms of their gender, location, age, education etc., and apply strategies based on these segments. Reinforcement learning is quite different from the above two categorizations.

Reinforcement learning is about mapping situations to actions to maximize a reward signal. Here, there's something called a learner, which decides on its own which actions to consider for maximizing its reward. The actions taken by the learner might affect not only the immediate reward but also the following situation. Trial error and delayed reward are major distinguishing features of reinforcement learning. For example, an adaptive controller is used in a petroleum refinery for adjusting parameters for optimizing cost, quality, and yield. The reinforcement learning system involves a policy, reward signal, value function and a model of the environment. This kind of learning behaves like a small child learning to walk in the initial phase and then advances as it starts interacting with its environment. The research on indoor localization utilizes a supervised learning approach, so the next section describes the process in detail.

3.2 Supervised Learning Algorithms

Supervised learning is further classified as parametric and non-parametric learning. The parametric learning model uses a fixed number of parameters, whereas the non-parametric model's learning is determined by data, and hence the number of parameters are defined by data. Supervised parametric learning is like a control system that has a fixed number of knobs where input data comes in, processed by how you turn the knobs, and output is a prediction. Turning knobs can lead to either proper/improper learning. The system proposed in the thesis is based on supervised parametric learning, where prediction of distance is based on how we turn on knobs of parameters like RSSI and locations. Target refers to an entity that is predicted. The properties of the target segregate the learning process into classification and regression. Dataset is defined in terms of input or observations and corresponding target with the assumption of real domain in which case we have x features/inputs/observations and y target.

3.2.1 Formulation of a supervised learning approach

$$\text{Dataset D} = \{(\boldsymbol{x_1}, \mathbf{y_1}),\ (\boldsymbol{x_2}, \mathbf{y_2}), (\boldsymbol{x_3}, \mathbf{y_3}) \ldots\ldots\ldots (\boldsymbol{x_n}, \mathbf{y_n})\} \quad \ldots\ldots\ldots \text{(Eq. 2-1)}$$

Where;

x_i € X → inputs/features/observations

y_i € Y → corresponding target.

Assumption made: X € R^m Where m → d-dimensional vector.

As described before, the properties and target result in the segregation of the supervised learning into regression and classification. Classification results in the generation of a classifier function whose cardinality is finite like binary labels, multi-class labels. In the Regression problem, we have output, which is continuous; possible examples include output/target (Y) belonging to a real domain R or [0, ∞). There's no clear line between classification and regression; we can transform a multi-label classification problem and formulate it as a regression problem. On the other hand, we can threshold the regression predictions and transform them into a multi-label classification by rounding the predictions to the closest integer. Modelling depends on prior information about a specific domain. Regression-based supervised parametric learning is a wise choice for indoor localization algorithms based on distance estimation using RSSI parameters. Supervised learning algorithms include both regression and classification algorithms (*Grokking Machine Learning*, no date).

3.3 Regression

The regression learning starts with a hypothesis in the functional form of the relationship

$$\boldsymbol{f(x) = w_0 + w_1 x_1 + w_2 x_2} \quad \text{.....(Eq. 2-2)}$$

Where;

w = w_0, w_1, w_2 are the parameters that need to be learned

And X = x_1, x_2 are the features of the dataset.

Searching for the best parameters w is known as a linear regression problem. Every other relation between feature and target falls under non – linear learning.

3.3.1 A simple form of Linear regression

The simplest form of linear regression is a straight-line relationship which estimates the regression coefficients as mentioned below –

$$\boldsymbol{Y_j = \alpha_1 x_i + \alpha_0 + \varepsilon} \text{ ..... (Eq. 2-3)}$$

Where;

Y is a dependent variable, X is the independent variable, α_1 is the slope, α_0 is the y – intercept and ε is the error.

Line fitting problem describes the relationship between feature and target considering slope, y-intercept, and error. Error is assumed to have zero mean and variance. When N observations are done using features and target, a method of least square estimates α_0 and α_1 as well as any possible hidden relationship. The precision of estimates describes how good the line will fit to data.

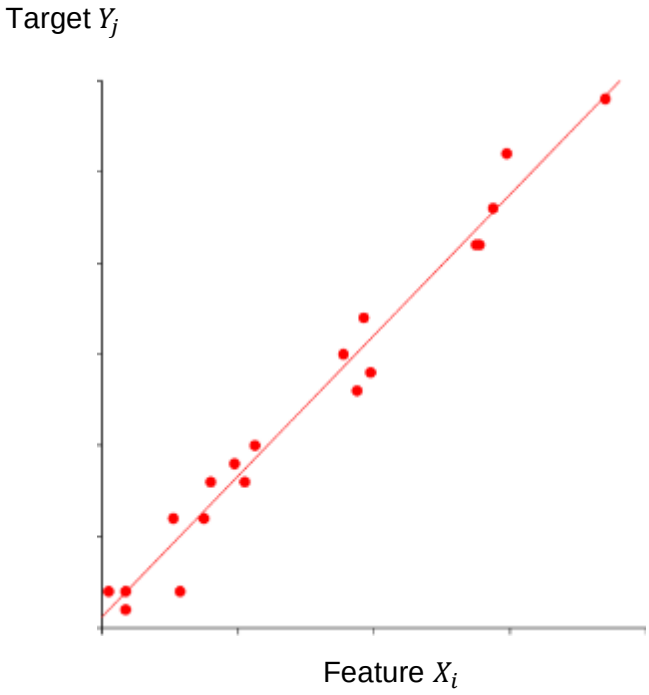


Figure 3-1. Linear Regression fit

The models generated form linear regression modelling include partial least squares and penalized models like lasso, ridge, and elastic net. These models seek to find parameter estimates to minimize the sum of squared errors or function of the sum of mean squared error. Their mathematical nature allows us to compute coefficients assuming distributions of the model residuals. There are certain limitations when it comes to linear regression models –

- The primary assumption is that relationship between features and targets lie on the same hyperplane, so if data had just one predictor, this modelling proves to be useful
- As the number of features rises, the relationship widens, and the linearity does not hold for such cases; in those cases, we need to augment features with additional functions of features and try to capture those relationships.

3.3.1.1 Best fits with Linear Regression

The aim of regression models is a prediction of a numeric target value. As described in the previous section, a more straightforward way of doing this is to write out a regression equation that relates targets to their inputs. Finding out regression weights or coefficients is what we term as regression. Once the regression weights are found, the process of forecasting becomes easy. Linear regression and regression terms are used interchangeably. Following outlines the overview of regression approach -

- Data Collection via a data source.
- Data preparation - only numeric values are used, and nominal values are transformed to binary values.
- Visualization of regression using any plot which relates targets to inputs .
- Computation of regression weights or coefficients also referred to as the training process.
- Using the weights from trained models on unseen data for forecasting and prediction also referred to as the testing/inference process.
- Measurement of accuracy or R-squared and correlation.

3.3.1.2 Pros and Cons of using Linear Regression

i. Regression works both with nominal and numeric values.
ii. Easy interpretation of results.
iii. Computationally inexpensive.
iv. Regression often results in poor modelling for non-linear data.

3.3.1.3 Least Squares Estimation

Estimation of the coefficient values can be performed using least-squares in the case of multiple inputs. The least-squares is aimed at minimization of the sum of squared residuals. The operation

is performed by calculating the distance from each data point to the regression line, then squaring it and summation of squared errors. The aim of squared error is to minimize this error. The operation is relatively simpler and computationally inexpensive.

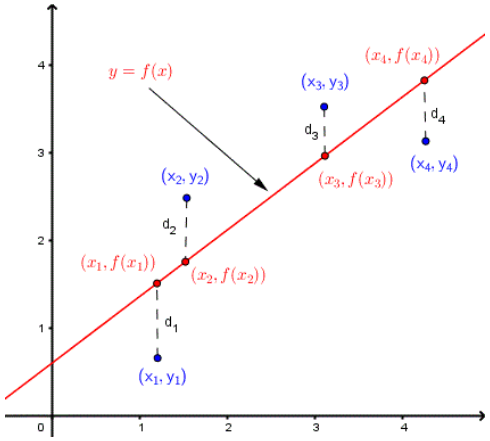


Figure 3-2. Least Squares Fit

3.3.1.4 Gradient Descent

Estimation of the coefficient values can also be performed using gradient descent which iteratively minimizes the error of the model during training of data. This method performs a summation of squared errors for each input-output value pair. As a scale factor, the learning rate is used, which is a hyperparameter. The optimization is done by iterative minimization of error. Learning rate determines the correction to be performed on each iteration of optimization (Robbins, 2007). However, this approach has an advantage when we have a large dataset.

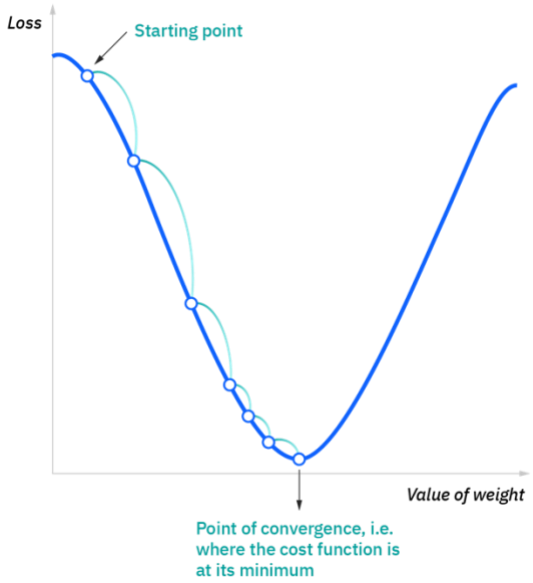


Figure 3-3. Gradient Descent algorithm

3.3.1.5 Overfitting and Regularization

A major issue with machine learning models is modelling error which occurs when a model fits a limited set of data points. The root causes of this error lie in bias and variance. Bias refers to assumptions a model makes to simplify the learning process for the target function. Low bias leads to fewer assumptions making the model complex for learning, and high bias leads to accessible models which quickly fits to a limited set of data points but disagree with unseen data points. Variance is defined by the variability of target function over different data points. Low variance causes target function to change at a low rate with changes in data, and vice versa holds for high variance. Linear regression is defined by the linear relationship between target and features, which often tend to have a high bias with low variance. To reduce overfitting, models are supposed to reduce bias and increase variance. It is possible to reduce overfitting by applying the following techniques -

i. Early stopping - During the iterative learning process of a learner, the training process can be stopped before the final iteration.
ii. Pruning - Commonly used in decision trees, tress can be stopped from growing by pre-pruning or post-pruning.

iii. Dropout - A widely applied technique where a randomly set of selected neurons are dropped during training.

iv. Cross-Validation - Splitting of the dataset into train and test data where training data is used for model generation, and test data is a phase where the model is used for prediction. A validation dataset is also used as a part of the training phase, where a fraction of the training dataset is used for the validation of the model. Cross validation may not reduce overfitting on its own but it gives a better insight on models which further helps in reducing overfitting. It allows models to generalize and prevents model to draw patterns from noise.

v. Regularization - Another popular approach for regularizing or shrinking the estimates of coefficients towards zero. This reduces the complexity of the model (Ying, 2019).

vi. Two of the common approaches are Lasso and Ridge regression. Lasso regression is used to modify ordinary least squares by minimization of the absolute sum of coefficients or what is generally referred to as L1-norm. Ridge regression is used to modify ordinary least squares by minimization of the squared absolute sum of coefficients or what is generally referred to as L2-norm.

3.3.2 Decision Tree Regression

A prime example of the divide and conquer strategy used in sorting algorithms. Decision trees are based on this divide and conquer strategy. The aim of learning is to figure out which questions to ask, decide the order in which these questions need to be asked and make predictions once we have enough questions to get a valid answer. When this set of questions are organized in a tree-like structure, the approach is referred to as a decision tree. CART or Classification and Regression tree algorithms is another name coined for decision trees. Decision trees are binary trees at the core level of implementation. Each node consists of an input variable and a split point. Leaf nodes are output variables used for making a prediction.

3.3.2.1 Decision tree representation

Each tree comprises nodes where each node is associated with one of the input variables and a split point on that variable. Edges represent possible values of that node. Leaf nodes constitute an output variable used for making a prediction. The following example represents a decision tree model -

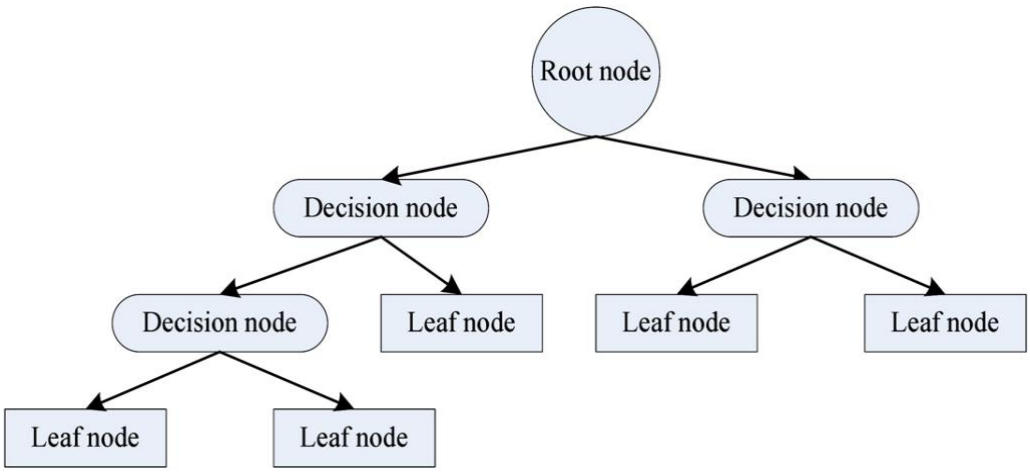


Figure 3-4. Decision tree representation

3.3.2.2 Predictions

The above model makes predictions by evaluating the input at the root node, deciding a split point based on values of this root node, partitioning the input space. Each input variable can be thought of as a dimension on an n-dimensional space. When the new data lands on a certain hyperplane of this n-dimensional space, it gets filtered, and the output value of this hyperplane is the prediction or target of the decision tree model.

3.3.2.3 Model learning

As described above, the prediction is a relatively simpler approach of iteration of binary decisions made at each step. Partitioning of the input space is performed by a greedy approach which divides the space referred to as recursive binary splitting. All values are gathered, and numerous split points are tried out and tested via the use of a cost function. The cost function evaluates the

split in terms of best or minimized cost and split them with the optimized cost is selected. The cost functions are described below -

- Regression modelling cost function - Sum of squared errors across all training samples is chosen as a cost function to evaluate the split point.

$$\sum_{i=1}^{n} (y_i - \boldsymbol{prediction}_i)^2 \quad \text{..... (Eq. 2-4)}$$

- Classification cost function - Gini cost function is used for indication of the n of purity of leaf nodes. This purity represents the blend of training samples assigned to each node. Gini cost of 0.5 represents the worst purity since G in the below equation will have 50-50 split.

$$\boldsymbol{G} = \sum_{i=1}^{n} \boldsymbol{p}_i \; \boldsymbol{X} \; (\mathbf{1} - \boldsymbol{p}_i) \quad \text{..... (Eq. 2-5)}$$

3.3.2.4 Criteria for stopping

A criterion or a stopping point needs to be set for recursive binary splitting; otherwise tree might never stop from growing. Systematic approaches include setting a minimum count on training instances assigned for each leaf node. Split won't be accepted if the count is less than criteria set and it will be rendered as a final leaf node. Pruning is a helpful technique for raising the performance of a tree. Occam's razor (Rasmussen and Ghahramani, 2000) says simpler hypotheses or models with fewer coefficients should be preferred to reduce complexity. Hence simpler trees should be given higher priority. The complexity of a CART or a decision tree is determined by the number of splits. A hold-out test set can be used as a rapid pruning method. However, removal of leaf nodes should be done only to ensure minimization of the cost function. Other approaches include alpha pruning, where the learning rate decides the weight of removal of nodes.

3.3.2.5 Pros and cons of using Decision Trees / CART

Pros

1. This method is not in need of any kind of data normalization or scaling.

2. Data imputation is not necessary as it doesn't provide any significant changes in the decision tree building.

3. No extra efforts have to be put in for data pre-processing.

4. An easy approach without inducing a complexity factor in modelling.

Cons

1. Not efficient implementation for regression and classification

2. Higher training time.

3. Computationally intensive.

4. Small changes in data can render the model unstable for prediction.

3.3.2.6 K - Nearest Neighbours

K-nearest Neighbours is another simpler supervised learning algorithm. Due to its low calculation and ease of use, KNN is widely used by the research industry. KNN classifies data points based on similarity. An educated guess is made on test data to classify unclassified points. KNN uses the dataset directly for prediction. K-most similar instances or neighbours are searched through the entire training set, and output variables are summarized for those K-instances. A distance measure is used for the determination of similarity with the K-instances.

3.3.2.7 Distance measure

Following are some of the distance measures used for KNN -

- Hamming distance - Distance between binary vectors is calculated. It refers to the number of bits positions in which the two bits are different. Primarily used for error correction or detection over data transmission networks (Norouzi, Fleet and Salakhutdinov, 2012).
- Manhattan distance - Distance between real vectors is calculated using the sum of their absolute difference. Also referred to as taxicab geometry or city block distance, the distance between two city blocks or two points in a grid is formulated as a grid path (Craw, 2017).

$$\boldsymbol{d} = \sum_{i=1}^{n} |\boldsymbol{x_i} - \boldsymbol{y_i}| \text{ ..... (Eq. 2-6)}$$

- Euclidean distance - The most famous distance measure used which calculates similarity in terms of squared roots of sum of squared differences between two points across all input attributes (Dokmanic *et al.*, 2015).

$$\boldsymbol{d} = \sqrt{\sum_{i=1}^{n} (\boldsymbol{xi} - \boldsymbol{yi})^2} \quad \text{..... (Eq. 2-7)}$$

- Minkowski's distance - A generalized distance metric that can be used with Euclidean or Manhattan distance formula. In short, it combines all L-norms in the form of Lp-norm where p = 1, 2,3, infinity etc (Çolakoğlu, 2019).

$$\boldsymbol{d} = \sum_{i=1}^{n} |\boldsymbol{xi} - \boldsymbol{yi_p}|^{1/p} \text{..... (Eq. 2-8)}$$

3.3.3 KNN for Classification / Regression

For Classification, KNN output evaluates the K-most similar instances and uses the highest frequency from those K-most instances for output. For Regression, it's the mean/median of these K-most instances that makes the prediction. Votes for each class are done based on each instance, and the class that has the most votes use used for prediction. The frequency of samples belonging to each class of K - most similar instance set is normalized, and probabilities of classes are evaluated.

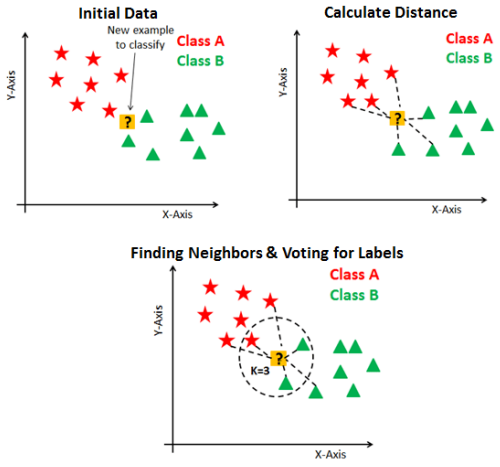


Figure 3-5. KNN classification

3.3.3.1 Advantages and Disadvantages of KNNs

Advantages

1. As seen, KNNS can be used for both Classification and Regression.
2. Numerous distance measures can be used for determining the similarity of instances.
3. No need of building a model.
4. Pretty simple and intuitive.

Disadvantages

1. Rather a slow implementation.
2. Sensitivity to outliers.
3. The curse of dimensionality is a major issue for KNNs.
4. Data imbalances lead to performance issues.

3.4 Classification Algorithms

A major fraction of machine learning algorithms is concentrated on solving a common classification problem. The learner is required to learn the hypothesis, which is used for mapping vectors into one of many classes by looking at output examples or labels of the hypothesis. This classification domain makes use of linear classifiers, Naïve Bayes classifiers, Logistic regression, K-means clustering, decision trees etc (Babcock University *et al.*, 2017). This section describes some of them in a concise manner.

3.4.1 Logistic Regression

The algorithm relies on the use of a single multinomial logistic regression model with a single estimator. It identifies the states where the boundary between classes exist and also formulates class probabilities based on distance from the boundary in a specific approach (Sengar, Gaikwad

and Nagdive, 2020). It is more robust at detailed predictions. It is one of the most commonly used tools for discrete data analysis and sometimes also referred to as linear interpolation. Regression is about prediction of a value for input based on past data whereas classification aims at classifying data points into different categories.

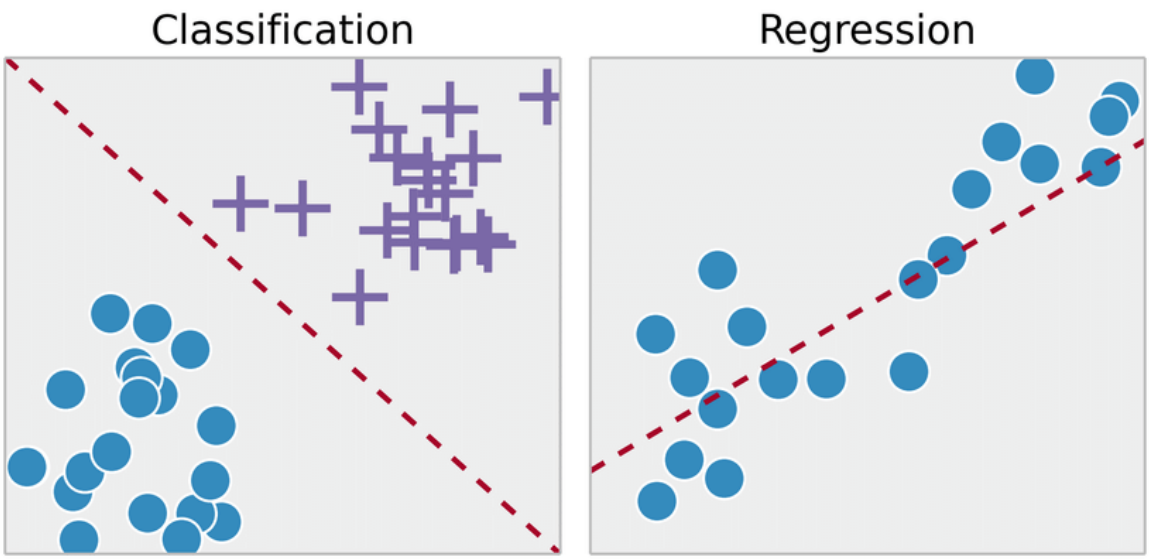


Figure 3-6. Classification vs Regression

3.4.2 Multi-layer perceptron (ANNs)

The algorithm is a classifier with weights whose values can be searched for by the use of a quadratic problem with linear constraints. In generic neural network training, the focus is on solving a non-convex unconstrained minimization problem. The classifier is generally used for learning from a batch of training instances iteratively. The process runs until a prediction vector is found, which is accurate on all of the training sets. This prediction vector is then used for the prediction of labels on a test set. These classes of algorithms can be used for both classification and regression. Artificial Neural Network (ANNs) make use of three fundamental elements, input and activation functions of each unit, network architecture and weight of each input connection(Kotsiantis, Zaharakis and Pintelas, 2006). The ANN's behaviour is defined by the current values of weights given the network architecture, and input connections are fixed. The weights are adjusted overall in order to reduce the error in the output values. The ANNs have been inspired by the human nervous system, which allows learning from representative data describing a decision process. An ANN consist of a layer of input node, layers of output nodes

and a single or more hidden node layers. The information or input is passed to hidden layers from input layers by firing of activation functions. The hidden nodes may fire or remain dormant based on the evidence. They apply weights to evidence in the form of function. When the values of these nodes reach a threshold, they are fired, and information is passed onto another layer and doesn't work efficiently for a tiny fraction of input data samples as they are not sufficient to train the model. ANNs are used to develop predictive modelling, optimization and control.

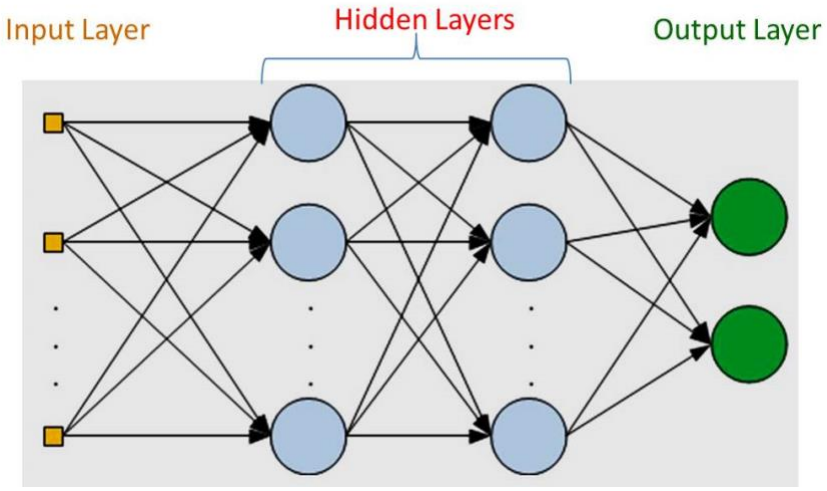


Figure 3-7. Artificial Neural Networks

3.4.3 Naïve Bayes Classifier

One of the most simple networks consisting of directed acyclic graphs with a single parent and several children nodes with the independence assumption amongst them. A model is rendered based on this independent assumption which is used further for estimation. This group of classifiers are less accurate than neural networks. Sometimes they prove to be better than other SOA algorithms like instance-based learning and decision tree induction. The focus here is on conditional probabilities, but not on the probability that something will happen, instead of the likelihood that an event might happen given that some other event happens. For example, we can try to classify emails; given an email, we would like to calculate the likelihood that it is spam. Bayes theorem is used for the calculation of conditional probabilities. Naïve Bayes assumes that given some input variables, they are assumed to be independent. The weaker part of the algorithm is that it performs better with categorical than with the numerical values, as it assumes the bell curve distribution. This assumption could sometimes harm the performance. The other underlying issue is the zero-frequency problem – if a categorical variable has some new category in the test set which might be absent from the training set, then the model will automatically assume it with

a 0 probability, and that won't be effective while making a prediction. This classifier is prominently used in spam filtering, image classification, document classification, sentiment analysis etc.

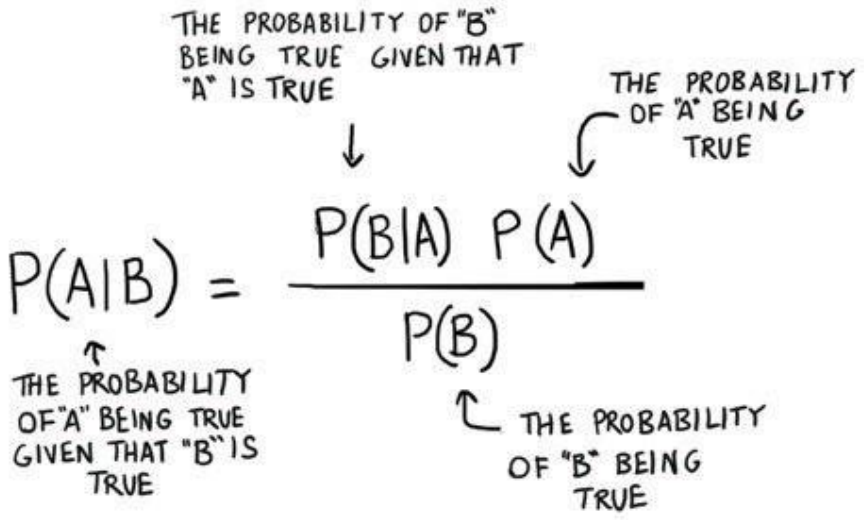


Figure 3-8. Bayes Theorem applied to Naive Bayes Classifier

3.5 Ensemble Approach for Weak learners

An ideal approach to the generation of better predictions is by combining predictions from several models. The core principle of ensemble algorithms is to group several weak learners to form a strong learner. The accuracy of the model can be increased with this approach. Models differ from each other in terms of variance, bias and noise. The ensemble approach helps remove the instability caused due to variance and bias. These minor differences lead to either overfitting or underfitting in models. The error gap gets significant between the training and testing error due to differences in bias, variance and noise. Ensemble algorithms are further classified into three subcategories – Bagging, Stacking and Boosting.

3.5.1 Bagging

Bagging or Bootstrap Aggregation is one of the powerful ensemble approaches. Bagging is a generic technique of reducing variance for high variance algorithms. The list of algorithms commonly influenced by high variance is decision trees. Decision trees are easily influenced by the training data. Infinitesimal changes in training data can influence the predictions made by the decision trees. In bagging, predictions made by several weak learners or models are combined using voting or averaging. Bagging reduces the variance of the data and prevents the models from overfitting. Bagging with decision trees is done by growing the trees deep instead of pruning, as this leads to low bias and high variance in predictions. Care has to be taken to ensure a unique

sampling of the dataset for each model(Brownlee, 2016a). Replacement is allowed, which means a row selected is included back in the training set for getting selected for the same training set. This is referred to as a bootstrap sample, used typically with small datasets for estimation of the statistical value of a data point. A better prediction is possible with the mean of estimates prepared using multiple bootstrap samples. In summary, bagging has three major elements –

I. Bootstrap samples

II. Unpruned decision trees.

III. Mean of predictions

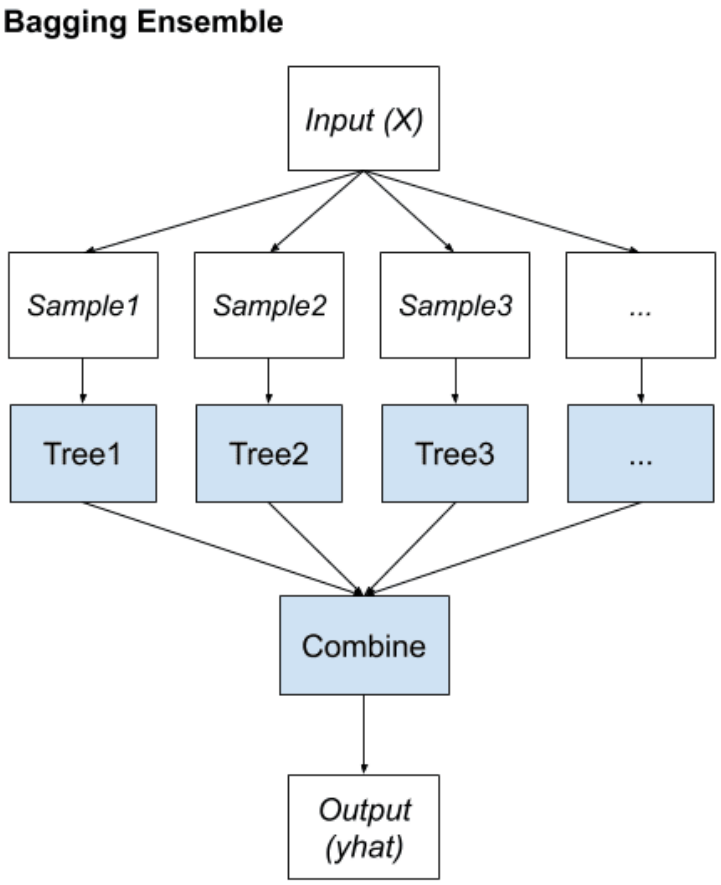


Figure 3-9. Ensemble models using bagging method

3.5.2 Stacking

An ensemble approach of learning combines regression or classification models via a meta-regressor or meta-classifier. The base model is trained on the entire training set, and the meta-model is trained on outputs of the base model trained using the outputs as features. The base model can be generated by different algorithms, and hence this form of ensemble learning involves heterogeneous ensembles. Stacking can make use of Naïve Bayes classifier, KNN or Random forests as a base model and logistic regression being used as a meta classifier(Brownlee, 2020). The stacking process is outlined below –

1. The training set is split into disjoint sets.
2. Base learners are trained on the above training set.
3. Base learners are tested on the second one
4. A high-level learner is trained using predictions from Step (III) as inputs.

Algorithm Stacking

Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
Ouput: ensemble classifier H
Step 1: learn base-level classifiers
for $t = 1$ to T **do**
    learn h_t based on D
end for
Step 2: construct new data set of predictions
for $i = 1$ to m **do**
    $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_T(x_i)\}$
end for
Step 3: learn a meta-classifier
learn H based on D_h
return H

Figure 3-10. Stacking Algorithm

3.5.3 Boosting

As the name suggests, this kind of ensemble transforms weak learners into solid learners using boosting. This approach is just slightly better than an educated guess on fitting weak learners.

Boosting helps in reducing bias and variance; examples include Ada-boost, Stochastic Gradient Boosting etc. The core idea is the correction of prediction errors. Models are made to fit and added sequentially to the ensemble in a way to correct predictions of the first model by the second model, second to third etc. Predictions are usually combined with averaging or simple voting(Brownlee, 2016b). No changes are made in the training dataset and algorithms are modified to make better corrections. The process of boosting is highlighted below –

I. Adding bias for samples that are hard for predictions.
II. Adding ensemble members iteratively for correction of predictions.
III. Weighted averaging predictions of models.

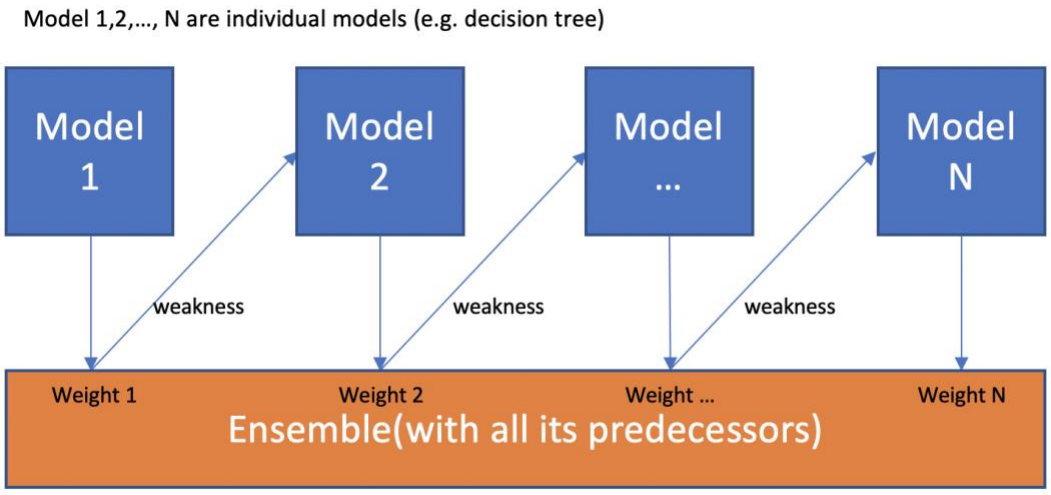


Figure 3-11. Boosting approach of ensemble learning

3.6 Comparison of fingerprint indoor localization approaches based on machine learning algorithms.

This section summarizes the use of different machine learning approaches used for fingerprinting based on RSSI and how each implementation is better than the previous one. The summary has been plotted in a tabular format as below –

Algorithm used	Fingerprint	Complexity	Accuracy	Important points

KNN	RSSI	Medium	2-3 m	Low accuracy as KNN is used for indoor localization. (Bahl and Padmanabhan, 2000).
Semi-supervised learning	Bluetooth	Medium	Not specified	Multiple signals fused to form a unified model, efforts of manual calibration are reduced. However, training and testing occur at a much faster rate (Jiang *et al.*, 2018).
Improved KNN	RSSI	Low	3-4 m	A better selection method in KNN (Tang, Deng and Huang, 2016)
NN	RSSI	High	4 m	Neural networks used for modelling and training non-linear Wi-Fi signal propagation(Dinh-Van *et al.*, 2017).
Ensemble Classification	RSSI	High	2.25 m	Constraints from the environment are used for solving the multipath effect with ensemble classification (Atia, Noureldin and Korenberg, 2013).
Bayesian	RSSI	Medium	2-3 m	Bayesian Kernel parameters updating is done constantly for calibration and estimation of radio maps (Guo *et al.*, 2018).
KNN	CSI	NA	1.203 m	Use of Euclidean distance measure for estimation of distance b/w the target point and reference point (Zhou *et al.*, 2017).
SVM	CSI	NA	1.22 m	PCA used for dimension reduction and then DBSCAN implementation for reduction of noise(Wang *et al.*, 2018)

CNN	Magnetic	NA	1 m	Magnetic field and RSSI integrated into an image and position classification done via the use of a convolutional neural network (Subbu, Gozick and Dantu, 2013).
ML	Visible light	NA	Less than 5 cm	Multiple classifiers are constructed for improving the localization performance (Zhao *et al.*, 2017).
RF,SVM	Bluetooth	NA	0.8-2.35 m	Dead reckoning and position fingerprints are fused to form a composite fingerprint for improving accuracy(Xiaodong *et al.*, 2018).
DTW	Visible light	NA	Less than 1 m	A navigation framework for the collection of light intensity fingerprints while walking reducing walking efforts (Wu *et al.*, 2013).

Table 3-1. Comparison of indoor localization fingerprint approaches

3.7 Summary

The review conducted gave a generalization of different supervised learning algorithms. Classification and Regression cover major use case including indoor localization. Regression is aimed at prediction or forecasting of continuous dependent variables from a number of independent variables. For the proposed framework, regression phase covers distance estimation using fingerprinting methods. The goal of classification phase is to predict class of given data instances/points. Decision Trees can be used for both classification as well as regression. The use of artificial neural networks can bring down errors following gradient descent algorithm. Ensemble machine learning involves use of multiple models which are weak learners but when combined together, they obtain more accurate and robust models (Rocca, 2021). Bagging is the mode of ensemble learning where base models are trained in parallel on several bootstrap samples and follow the averaging process to obtain strong model. Boosting is the mode of ensemble learning where base models are trained sequentially and learning of current weak

learner is dependent on previous weak learners. Boosting prefers weak learners with high bias and low variance. Stacking is the mode of ensemble learning where several weak learners are fitted independently. Training of a meta-model ensures prediction of outputs based on outputs returned by base models.

Chapter 4

Experimental Setup

4.1 Introduction

This chapter covers details on dataset generation and model generation for indoor localization framework. It introduces a brief summary of the experimental setup, including the use of proprietary hardware and software along with cloud infrastructure. It also highlights the use of some standard datasets along with the standard machine learning pipeline design. The execution involves the use of open-source tools including Python3, standard python and machine learning libraries and the use of a low code framework. The chapter starts with a brief dive into mesh architecture used for sending fingerprint (RSSI) data onto cloud. The cloud architecture is a standard PaaS (Platform-As-a-Service) platform which suites current research platform. The main reason of using this platform is because of its simplicity, cost-effective development, scalability and high availability. The proposed framework supports local development without reliability on any cloud architecture details of which are covered in the appendix. The later part of this chapter describes the dataset generation and model generation algorithms.

4.2 Experimental Setup

This section describes the architecture setup involving Bluetooth mesh setup via the use of three primary components - Beacons, Servers and Gateways. The experimental setup described in this section is limited to the scope of this study. The experiments conducted makes use of water leak detection sensors which are BLE-based and hence makes use of RSSI fingerprint. The reason behind choosing this novel leak detection sensor is low equipment cost, low power, and easy setup. The setup also includes a novel Bluetooth mesh systems using energy harvesting powered beacons to send notification signals which mesh devices can identify and send over the mesh, reaching endpoints. The combination of Beacons and Mesh (referred to as Beacon Mesh Integration system) sends out a BLE beacon while they are powered. It is not necessary to use the same architecture while testing out machine learning frameworks in different domains. The setup can be replaced with Wi-Fi or another fingerprint setup. Bluetooth mesh extends the capabilities of the current BLE to include mesh networking. A many to many communications network can be extended to thousands of devices. Reliability is a significant part of Bluetooth mesh which is ensured with multiple paths from source to destination without a single point of

failure. The design has been driven with an aim to create an efficient, flexible and simple wireless mesh networking solution. A message entering the mesh network can is forwarded by multiple relaying nodes. Relaying is possible because of a flooding communication model. For hardware, we are relying on Nordic semiconductor's products for the BLE mesh stack. The following underlines our current mesh architecture.

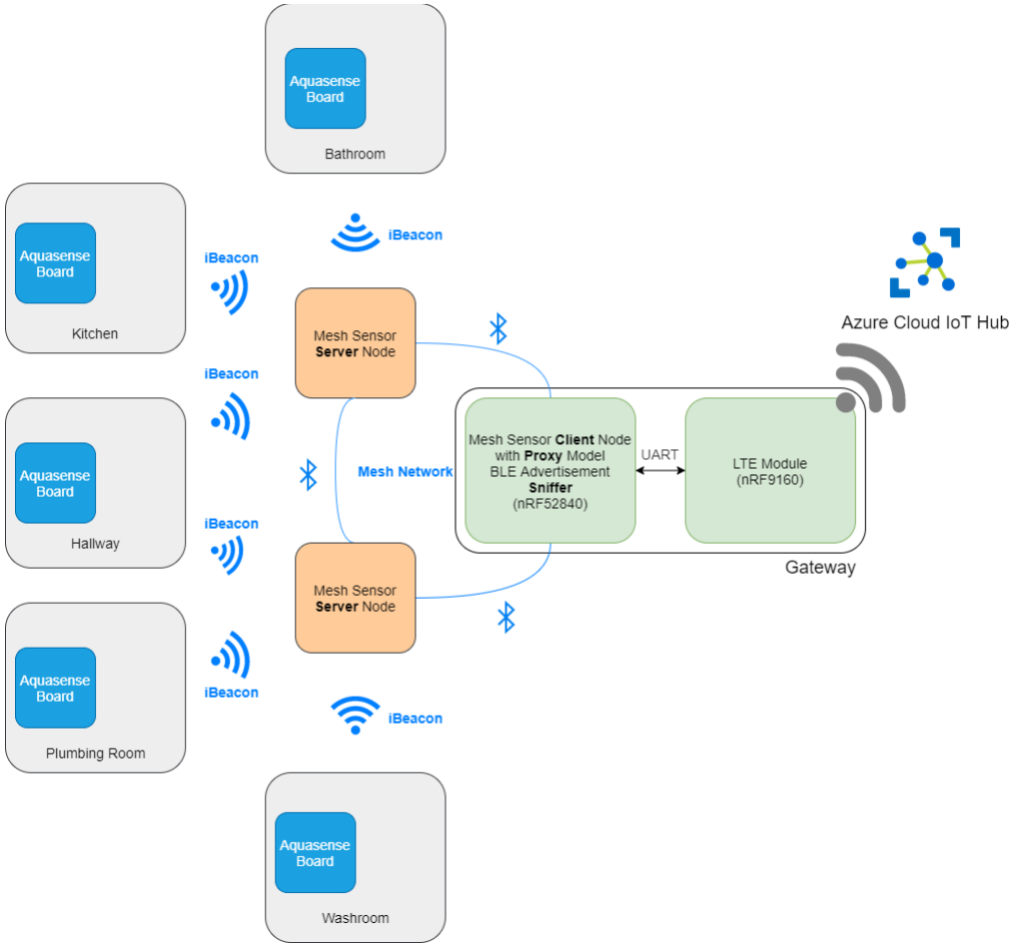


Figure 4-1. Mesh network implementation using Nordic semiconductors

A custom BLE board is designed for being used as a Beacon. The Beacon connects to an energy harvesting board and water leak sensor. A plastic housing case has been designed using CAD. When in contact with water, the beacon's underlying magnesium and graphite sensor power Nordic semiconductor's nrF52832 chip. Once turned on, this sensor sends packets to the mesh network while it is powered. The messages being transmitted by the beacons contain the following information - Sensor address (128-bit hexadecimal address for identification of Bluetooth

address), Mac address (sensor's MAC id), RSSI (fingerprint values) and a GUID for gateway identification.

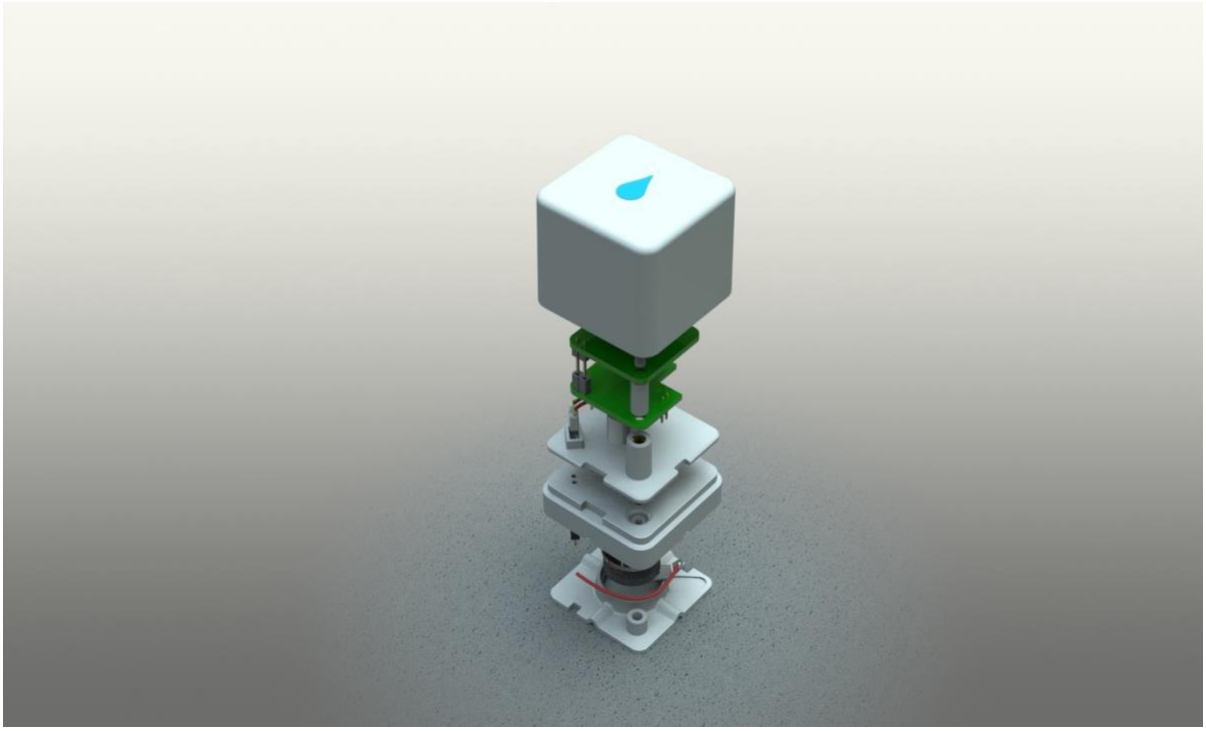


Figure 4-2. Expanded view of leak sensor for RSSI fingerprint measurement

Servers rely on Beacons which are the same custom BLE boards utilizing the nrF52832 chip. A DC board powers the server using a wall outlet. These boards are always powered on and ready for transmission of packets it receives from other beacons in the mesh network. Relaying packets from one server to another server is an important feature. Multiple hops are possible for packets between beacons between servers and then be received by the gateway. These packets are then further sent to the cloud for data processing.

Gateway uses the Fantsel BLG,840x, which is an industrial gateway responsible for sending information from mesh to the cloud. This gateway is again a combination of nrf52840 and nrf9160 onboard, where the first one, i.e., nrf52840 is responsible for communication with the Bluetooth mesh, and the second one, i.e., nrf9160 is an LTE module responsible manufactured by Nordic responsible for communication between gateway and cloud. Intercommunication between these two modules is done via use of UART connection. The client refers to the firmware

running on the BLE side and gateway refers to the firmware running on the LTE side(Witham *et al.*, 2019).

Figure 4-3. BLG840x acting as a gateway for sending data to cloud.

4.3 Software

This section describes the software architecture used in the research. The section is further subdivided into following categories:

1. Cloud reference architecture
2. Data ingestion
3. Reference Dataset
4. Dataset generation
5. Machine learning pipeline

4.3.1 Cloud Reference Architecture

The azure platform is used as a PaaS (Platform-as-a-service) for our implementation. Most of the IoT use cases are fundamental processes of sending data in order to generate insights. These

actions can then be used for improving the process of indoor localization. The reference architecture being used is as shown below:

Figure 4-4. Azure IoT Reference Architecture.

The reference architecture (doodlemania2, no date)is composed of following elements (some of which are optional):

1. IoT devices - refers to devices that can connect with cloud securely and proceed with transmission and reception of messages. Edge devices can be used for performing data processing on devices for which use of Azure IoT Edge is preferred. In our use case, leak detection sensors are the core IoT devices used for the experiment.
2. Cloud gateway - A bidirectional gateway is acting as a hub for devices to connect to cloud for data transmission and reception. IoT Hub is one of the prime components being used for receiving data from the Nordic Bluetooth gateway and relaying the information further. IoT Hub makes provisions for device management, capabilities to control and command devices. The IoT Hub is responsible for getting data

ingestion from devices, thereby acting as a message broker between backend services and devices.

3. DPS or Device Provisioning Service - helps in registration and assignment of devices to IoT Hub endpoints at scale.
4. Stream Processing - As the name suggests, this element deals with streaming of extensive data records and applies rules to those streams in order to be processed further. Complex analysis can be executed at scale using stream aggregations, windowing functions and external data source joins.
5. Azure storage - can be used via two different pipelines. Cold path storage can be used for storing long term data, and this is generally used for batch processing, whereas warm path storage holds data for high availability for reporting and visualization in the form of dashboards via the use of business intelligence tools.
6. Logic apps - After gathering insights, the next step in the process is to drive actions which can be done via use of logic apps where actions can be included for storing messages, raising alarms, sending emails or SMS messages or integration with custom relationship management tools. Logic apps in our use case are used for raising alerts via use of mail connectors for sending emails to end-users.
7. Machine learning (Auto-ML) - makes use of several algorithms for prediction modelling. Models generated can be deployed and used over real-time data for making predictions to enable scenarios for maintenance or for localization.
8. Power Apps and BI platform - plays a vital role as a User Interface for representation of actionable insights or data which could be driving market for several businesses. The agnostic framework makes these tools highly useful in case of any kind of data related issues.

4.3.2 Data Ingestion

The reference architecture described above is used as a core implementation for data ingestion to be processed for further use. Following outlines, the process of data ingestion for collecting RSSI data from beacons via use of Nordic Bluetooth gateway -

a. The data from leak detection sensor is sent to Azure IoTHub via gateway via the use of a mesh network.

b. On the Azure cloud side, IoTHub has a default built-in endpoint that is also compatible with event hubs. The default topic on IoTHub is messages/events. Custom endpoints can also be used for routing messages to another service.
c. IoTHub supports various message protocols, including HTTPS/AMQP/MQTT etc. There are two ways to send data to the cloud using IoT devices -
    i. Registration of device with IoTHub using a security token/certificate.
    ii. Creation of virtual IoT device on IoTHub and using that device with SAS or Security Token with a TTL (Time to Live)

a. Following any one of the approaches, we see device activity in the IoTHub. Monitoring the data is possible via the use of Azure command-line interface.
b. Once the data packet reaches the default endpoint (messages/events), a similar streaming job ensures to store it in a database.
c. Azure SQL server database ensures the high availability of our data.
d. This data is stored in real-time and can be accessed via running SQL queries on the database or via the use of a script to access the table.
e. A logic app runs sequentially to send alerts to end-users regarding the leak detection or data collected from sensors/beacons.

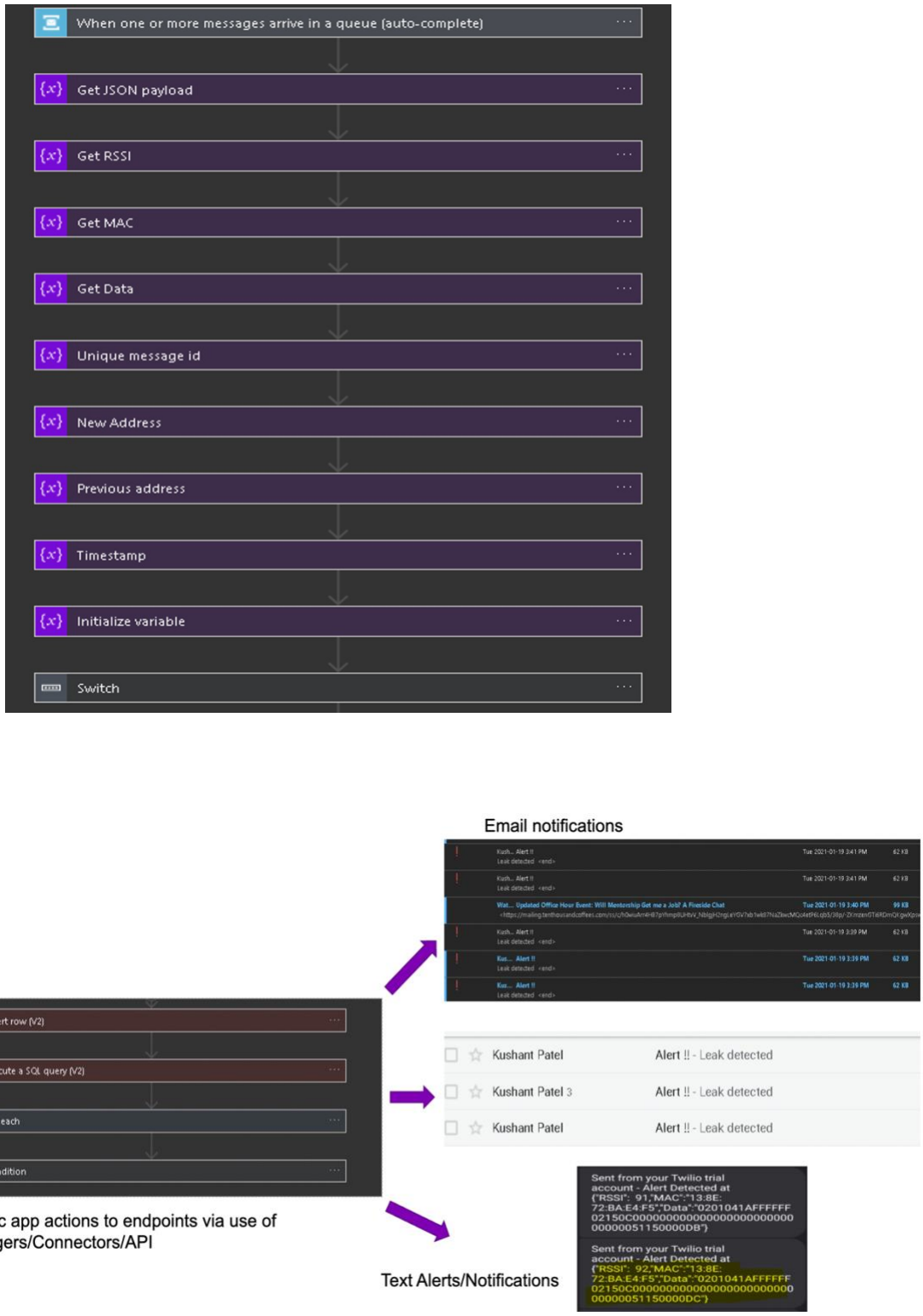


Figure 4-5. Cloud application for data acquisition and sending alerts to endpoints

4.3.3 Reference Dataset (Used for framework evaluation)

This section describes one of the standard reference datasets that was chosen for the evaluation of the algorithm. The dataset is the one used for indoor location determination with RSSI values (Malekpour, Ling and Lim, 2008). As a part of the research, 120k RSSI samples from 4 access points on floors, 1 and 2 of the building of the author's faculty have been taken (*Indoor location determination with RSSI*, 2018). A simple algorithm was used for finding the location of a Wi-Fi device in that building with 85 % accuracy (Malekpour, Ling and Lim, 2008). Following describes the outline of the dataset used-

a. Multiple samples are taken for each coordinate.
b. Access points are referred with A, B, C and D.
c. Physical coordinates are identified by grid coordinates (X, Y and Z coordinates), where X and Y are grid coordinates, and Z refers to the floor coordinates.
d. Signal strength fluctuation is prominent in any indoor location. Hence multiple samples are considered.
e. Each row of the sample dataset has the format -
    i. Ap, signal, sequence, X, Y and Z.
    ii. Ap - refers to the access point identifier (A, B, C or D).
    iii. Signal- refers to the signal strength from the access point.
    iv. Sequence - The sequence of the sample from any access point to a specific coordinate.
    v. x, y and z coordinates are the grid coordinates.

This dataset has been used as a benchmark for the evaluation of the framework.

4.3.4 Fingerprint Dataset generation

This section describes how to use any fingerprint and transform it from raw data to an organized location dataset for further evaluation of the framework. Dataset generation is one of the widely researched fields for machine learning purposes. Supervised machine learning algorithms need a labelled dataset for target inference. There are three main approaches used for the generation of labelled training datasets -

- Manual labelling
- Synthetic labelled dataset
- Semi-automatic labelled dataset

Manual labelling of the dataset is essential when we have no control over the environment during the collection of labels for dataset generation. Indoor localization is a challenging field where the environmental conditions keep on changing. There is no control over the interferences which get induced during the training and deployment phase. The fingerprint collected using manual labels is not an easy task. Various methods are employed for the collection of manual data. However, these manual labels are referred to as ground truth collected with the help of a domain expert. Sometimes, the whole task of gathering manual data can be tedious even for the domain experts and hence we can avail the use of synthetic dataset generation process. Synthetic dataset generation involves scripts programmed by humans. There are five steps involved in process of synthetic dataset generation process -

- A simulation architecture is defined, implemented and executed.
- During the simulation execution, data is collected.
- Relevant features are used for the generation of the dataset in this step.
- The result of the above three steps is split according to a specified proportion/fraction.
- A fraction of the dataset section is used for training the machine learning model, and another fraction is used for the evaluation of the training model.

The last approach utilizes a semi-automatic process of dataset generation. This approach is usually a blend of manual label generation and implementation of logic on labelling. Complex algorithms and heuristics can be used as a part of the labelling logic. This logic drives an automatic process of generation of dataset. The last two steps are similar to the ones used in the synthetic dataset generation process. Fingerprint dataset generation has been implemented by use of a semi-automatic approach. Following algorithm outlines the whole process-

Algorithm 1: Fingerprint Dataset Generation

Data: Raw fingerprints from sources (SQl server/ CSV file / JSON file / REST API)

Result: Labelled Dataset

Initialization;

Creation of Virtual Environment;

Installation of required packages and tools;

Manual Phase:

1. Get a floor plan for the indoor area you want to localize.
2. Acquisition of data with location coordinates manually using grid floor plan or geographic coordinate system (optional).
3. Adding a timestamp to the data entries
4. Storing the data onto a specified source.

Initial Phase:

1. Identify the source of data.
2. Load the Dataset (Ground Truth).
3. Identify feature columns and its types.
4. Type conversion of Numerical and Date/Time columns.
5. Addition of a fingerprint source address column (unique identifier).
6. Addition of an standard intensity scale.
7. Addition of location columns in form of grid/geographic .coordinate system (optional).

Labelling Logic:

1. Count the number of Access Points.
2. Enter the location column names (grid/geographic) coordinate system.
3. Append the location of each access point to access point list.
4. Use the Minkowski's generalized distance measure to estimate the distance of fingerprint source from Access Points.

$$d(f,a) = (\sum |f-a|^p)^{1/p} \quad (1)$$

where (f,a)$\rightarrow$ $(fingerprintlocation, accesspoint-location)$

and p = 1 for Manhattan distance or p =2 for Euclidean distance

Output: Fingeprint Dataset;

Figure 4-6. Dataset generation using a semi-automatic approach.

The following diagram represents a snapshot of the dataset generated using the above algorithm. CSV files have been used as a source along with RSSI as a fingerprint. However, this algorithm has been generalized to be used for any other fingerprint from any relevant source. The labels generated using this algorithm use Minkowski's distance metric, generalizing Euclidean and Manhattan distance metric for estimation of distance using data points.

	RSSI	MAC	Data	message_id	EventEnqueuedUtcTime	client	Sensor	RSSI_Intensity	location	lat	long	floor	distance
0	-62	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	0fb5857629fe4843bdcb6ae413246723	2021-06-03 17:27:40	aquasensing	Sensor Device	Very good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
1	-66	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	a9b1524bdeb841009bf1358727908b15	2021-06-03 17:28:35	aquasensing	Sensor Device	Very good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
2	-74	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	0f49dad24b1346589ae010a6039b1ae1	2021-06-03 17:28:36	aquasensing	Sensor Device	Good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
3	-74	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	d4a2405de0fe4ad397b292cb7d4d94ab	2021-06-03 17:28:39	aquasensing	Sensor Device	Good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
4	-66	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	10417258b298479afb6b364255dc215	2021-06-03 17:28:39	aquasensing	Sensor Device	Very good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
5	-73	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	77d053a8c192411991d143d19188ce99	2021-06-03 17:28:40	aquasensing	Sensor Device	Good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
6	-62	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	1574350bbe424ebe9227989a6753c9f6	2021-06-03 17:28:41	aquasensing	Sensor Device	Very good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
7	-77	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	c7ce7eee3c8f4eaea27e52f4c6fe7093	2021-06-03 17:28:43	aquasensing	Sensor Device	Good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
8	-65	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	7057a0d4cc9d4f5f822fe24c0ccb9b2d	2021-06-03 17:28:44	aquasensing	Sensor Device	Very good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756
9	-62	F4:B2:51:69:55:80	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55EE...	1999a298c9f74f70adfd02e88a5a58b5	2021-06-03 17:28:44	aquasensing	Sensor Device	Very good	Kitchen Counter (island)	43.4754	-80.5355	3	0.029756

Figure 4-7. Preview of the transformed dataset generated using Dataset generation algorithm.

4.3.5 Machine learning pipeline

Machine learning pipelines are becoming more critical in the field of self-driving cars, computer vision and predictive maintenance industries. Experts are necessary for deploying a pipeline which takes in the existing raw data and transforms it into a model which can be used for further prediction/classification purpose. In this section, we describe the architecture as well as the experiment conducted using the dataset transformed in the previous section.

4.3.5.1 The architecture of the ML pipeline

A machine learning pipeline transforms a feature vector into a target value with a sequential combination of various algorithms. Following represents the generic architecture of an ML pipeline -

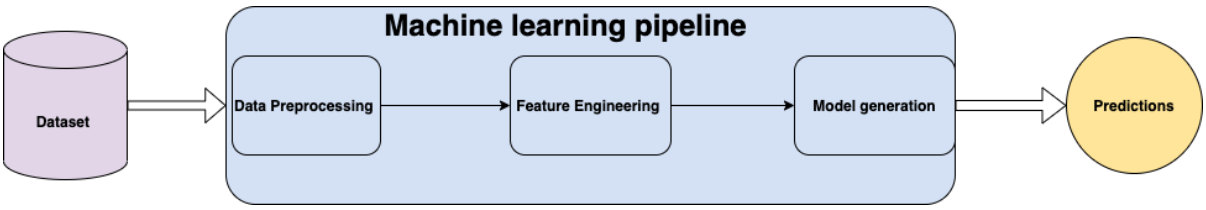


Figure 4-8. Machine learning pipeline architecture

4.3.5.2 Dataset

The dataset generation algorithm takes care of the transformation of the raw dataset. However, the transformed data might still be insufficient for regression and classification. There might be

some datatype conflicts, some feature imbalances, some imputation issues. The dataset loaded must have enough features correlated with your target. The indoor location dataset has a spatial context integrated with the dataset in the form of geocoordinates along with the target distance estimation. Details of the dataset are as follows -

4.3.5.3 Reference Dataset details

Indoor Location Determination with RSSI (120k RSSI samples to determine the distance in a two-level building) -

a. Four access points have been used for sampling RSSI levels for each row in a dataset
b. Access point identifiers are one of A, B, C and D locations
c. X, Y, Z are grid coordinates recorded for the experiment
d. Signal strength is the RSSI signal strength from each of the access points.
e. Locations of the access points are static and do not vary during the experiment - {"A": (23, 17, 2), "B": (23, 41, 2), "C”: (1, 15, 2), "D": (1, 41, 2)}
f. Wi-Fi radio signals have been used in 2.4GHz frequency, which gives an interesting approach to learn how signal varies indoors.
g. The major challenge was to come up with a machine learning model that estimates distance and also makes estimations of X, Y and Z coordinates.
h. Distance estimation is performed using a Euclidean distance equation.
i. The execution phase conducted by Author consisted of 120k samples which ran in 7644.6 seconds without the use of any accelerator.

4.3.5.4 Transformed Dataset details

Indoor location determination with RSSI (Approx. 2048 RSSI samples for RSSI intensity range level classification) -

A. Two access points have been used for sampling RSSI levels for each row in a dataset.
B. Access points have names in the form of actual locations indoors.
C. X, Y and Z coordinates are geocentric coordinates in the form of latitudes, longitudes and floor location, thereby having a spatial aspect to them.
D. Signal strength is still the RSSI signal from each of the novel leak detection sensors described in the hardware section, along with the cloud gateways.

E. A better network architecture for dataset acquisition.
F. Distance estimation is done using generalized Minkowski's distance equation.
G. Execution phase - to be completed.

4.3.5.5 Data preprocessing Phase

Preprocessing phase is an indispensable step when it comes to architecting a machine learning pipeline. This section describes some of the necessary preprocessing steps performed with both Reference and Transformed datasets in order to be used for classification and regression. Pycaret (*PyCaret — pycaret 2.2.0 documentation*, 2021), which is a low code machine learning library, has been used from this phase and onwards. The preprocessing phase includes -

- **Sample and Split** - The main aim of any machine learning algorithm is to build a model which generalizes well to new data. Hence, the dataset is often split into train and test datasets for supervised machine learning tasks. The test dataset here serves as a proxy for real-world data, which the deployed model will be exposed to in real-time for prediction or classification. Cross-validation ensures the overfitting of the model. For our use case, k-fold cross-validation has been used on the training set, and the test set is used as a holdout set which is used for prediction. A sampling of the dataset is enabled by default when the dataset contains more than 25k samples.
  (Parameters - train_size, default = 0.7)
- **Data Preparation** - This section is subdivided into further six subcategories with experiments on different preparation steps -
  - Missing Values imputation - Deals with the missing values or empty records in the dataset. Algorithms are themselves incapable of performing imputation; hence this step becomes necessary.

  (Parameters -
  numeric_imputation, default = 'mean' = missing values replaced with the mean value of the feature.
  categorical_imputation, default = 'constant' = missing values imputed with a constant 'not_available' value).
  - Changing Datatypes - Pycaret's inference algorithm does detection of the data type of each feature, and if the detection proves to be incorrect, a default approach is enabled to change the feature.

  (Parameters -

numeric_feature , default = None
categorical_feature , default = None
date_feature , default = None
ignore_feature , default = None).

- One - Hot Encoding - Commonly known as dummy encoding, works with the transformation of categorical features into numerical enumerated values before training a model.
- Ordinal Encoding - There might be variables in a dataset with natural intrinsic order such as Low, Medium and High, which have to be encoded differently than others.

  (Parameters -

  Ordinal_features:dictionary , default = None).
  - Cardinal Encoding - There might be variables in a dataset with many levels or high cardinality features; one hot encoding might become slow in such cases and also introduce noise; hence their conversion can be handled using high_cardinality_feature in pycaret.

  (Parameters -

  High_cardinality_features, default = None)

- **Scaling and transformation** - Normalization and transformation are part of preprocessing phase, where the goal is rescaling of numeric column values without losing information or value ranges. Z-score and min-max scaling are two popular approaches when it comes to normalization. Transformation is more of a distribution shape change where data must be transformed when machine learning algorithms are used. The data can be represented by a normal or approximate normal distribution instead of relying on gaussian and normality in residuals. Yeo-johnson and quantile are two popular approaches used for transformation.

  (*Parameters –*

  Normalize_method, default='zscore' (others include min-max, maxabs and robust, Transformation_method (string/bool), default = False for book and 'yeo-johnson' for string.).
- **Feature Engineering -** This step deals with the creation of new features from existing data for training a machine learning model. It includes feature interaction, creation of

polynomial, trigonometric or group features, binning numeric levels and combining rare levels.

(Parameters -

Feature_interaction: bool, default = False

Feature_ratio: bool, default = False

Interaction_threshold: bool, default = 0.01

Bin_numeric_features: list, default = None)

- **Feature Selection** - This step includes feature importance, removal of multicollinearity, principal component analysis and ignorance of low variance.

  (Parameters -

  Feature_selection: bool, default = False

  Feature_selection_threshold: float, default = 0.8

  Removal_multicollinearity: bool, default = False

  Pca : bool , default = False)

4.3.5.6 Model generation phase

This section describes the model generation phase via the use of the framework. There are two crucial classes of algorithms used in this section -

- Regression - This step is used for distance estimation using various regression algorithms.
- Classification - This step is used for intensity level classification using various classification algorithms.

The following image outlines the algorithm for a model generation -

Algorithm 1 Model Generation

Preprocessing Phase

1. Initialization.
2. Creation of Virtual Environment.
3. Installation of required packages and tools.
4. Select the fraction of data to be used for modeling and predictions.
5. Data pre-processing for Cleaning,Normalization,Transformation,removing multi-collinearity.
6. Feature engineering(Imputation, outlier detection etc.)
7. Dimension reduction for lower representation of numerical features.
8. Selection of Classification or Regression for use case.

Regression Phase

1. Select the distance to be estimated as target feature.
2. Split the data set into training and test sets.
3. Compare models using framework with eighteen supervised machine learning algorithms on train set.
4. Evaluation of model using evaluation metrics [MAE,MSE,RMSE,R2,MAPE and TT (sec)].
5. Select top three models based on above evaluation metrics.
6. Hyper parameter tune these three models using five fold training splits.
7. Creation of an Ensemble of models using bagging and boosting methods.
8. (optional) Blending or Stacking of models to create a final model.
9. Use the model generated for prediction of distance on hold out validation data set.
10. Calculation of R2 score using this model.
11. Make predictions using the generated model on unseen data set.
12. Check for final R2 score of the model on unseen data of data set.
13. Go to Deployment Phase.

Classification Phase

1. Select the RSSI intensity levels to be classified as target features.
2. Split the data set into training and test sets.

3. Compare models using framework with eighteen supervised machine learning algorithms on train set.
4. Evaluation of model using evaluation metrics [Accuracy,AUC,Recall,Precision,F1,Kappa,MCC,TT(sec)].
5. Select top three models based on above evaluation metrics.
6. Hyper parameter tune these three models using five fold training splits.
7. Plot the confusion matrix to check for accuracy of a classifiers.
8. Generate a classification report , Prediction Error , Decision boundary

9. Use the model generated for prediction of distance on hold out data set.
10. Make predictions using the generated model on unseen data set.
11. Check for accuracy.
12. Go to Deployment Phase.

Deployment Phase

1. Save the final model onto cloud or local machine.
2. Deploy this model for real time regression or classification once data set has been transformed.
3. Use deployed model as an API on cloud infrastructure to be generalized on unseen fingerprints for the same indoor structure.

Figure 4-9. Model generation phase.

4.4 Conclusion

This concludes the experimental setup section. The setup covered all the necessary aspects of fingerprint dataset generation. Any kind of fingerprint with a fraction of spatial context is useful to dataset generation. The transformed dataset is then further loaded onto machine learning pipeline which opts the algorithms best suited for a specific domain and makes predictions using the trained models. The cloud is an indivisible part of the framework. Data ingestion and reliable cloud servers make it easy to conduct efficient research. The endpoints make it easy to inform the end users with the leak alerts and also send email notifications with the necessary information regarding the leaks detected.

Chapter 5

Experimental Results and Analysis

5.1 Introduction

This chapter presents the results from the use of algorithms from the previous chapter. The results are based on the use of reference datasets and datasets generated via the use of dataset generation algorithms. The chapter is subdivided into two sub-sections - Distance Estimation using Reference dataset and Multilabel fingerprint intensity classification. The first section mentions the use of reference datasets (RSSI dataset) and the use of a new dataset generated for distance estimation. The last section describes a multi-label classification approach for fingerprint intensity. The assessment and evaluation of all results are done via the use of standard metrics used for evaluation of classification and regression models. For analyzing the performance of regression models, we use residuals plot, prediction error plot, cooks distance plot, learning curve, manifold learning, feature importance and model hyperparameters. For analyzing the performance of the classification model, we use AUC, Confusion Matrix, Threshold, precision-recall, error plot, class report, feature selection, learning curve, validation curve, feature importance and decision boundary.

5.2 Introduction to evaluation of machine learning models.

5.2.1 Regression Evaluation

Various kinds of plots can be used for model analysis. However, for our use case, three major types of plots have been used for the evaluation of a regression model. They are described as follows -

- **Residual plot** is simply a plot that denotes how much a regression line misses a data point vertically. The residual values are plotted on vertical axis and the independent variable are plotted on the independent variable. For a good plot, the residual values are randomly and equally spaced around the horizontal axis.
- **A prediction error plot** is a plot that shows the predicted values generated by the model against the actual targets from the dataset. This plot allows us to see how much variance is present in the model. Regression models can be compared against a 45-degree line which shows how the prediction matches the model

- **Learning curve plot** compares the performance of a model on training and test datasets over various training instances. This allows us to verify if a model is learning much about the data. A bad learning curve usually can be a possible effect of high variance and high bias. An ideal learning curve is a model that generalizes well to new data.
- **Other metrics include** mean absolute error, mean squared error, and root mean squared error.

5.2.2 Classification Evaluation

Evaluation of a good classifier naturally depends on its accuracy to classify a test data point accurately by its class. Different metrics are used for evaluation of classification models. The selection of an evaluation metric depends on the nature of the problem. Following describes some of the standard metrics used –

- **AUC (Area Under Curve)** is one of the frequently used evaluation metrics. The metric plots out sensitivity and specificity for each decision cut-off between 0 and 1. A threshold is used for the conversion of probability outputs to classifications. The AUC plots the true positive rate versus false-positive rates for each possible threshold.
- **Confusion Matrix** is used for the visualization of the performance of a model. Actual targets are plotted against the predicted ones. True positives, True negatives, False positives and False negatives are four elements that tell us about the performance of a classifier.
- **True Positives** are the ones in which we predict true, and the actual target is true.
- **True Negatives** are the ones where we predict false, and the actual target is false.
- **False Positives** are the ones where we predict true and actual target is false
- **False Negatives** are the ones where we predict false and actual target is true.
- **Accuracy** shows us how frequent is our classifier correct. It is the ratio of the number of correct predictions to the total number of predictions
- **Recall** presents the fraction of correct positive identifications out of all positives.
- **Precision** denotes the fraction of correctly identified positives out of all positive predictions.
- **Sensitivity** presents a model's evaluation of positive predictions for all categories.

- **Specificity** presents a model's evaluation of negative predictions for all categories .

  Most of these evaluation metrics alone are not enough to tell us how good a classifier is. Sometimes we need to optimize on a specific metric depending on the domain and the use case in which the classifier is trained and tested.

5.3 Distance Estimation (Regression Phase)

This section describes and analyses the regression phase used for distance estimation. Two datasets have been used for this purpose –

5.3.1 Reference Dataset based Euclidean distance estimation analysis.

To evaluate the accuracy and performance of the proposed framework, a reference dataset has been used. The indoor space used for this dataset is a two-level building with four access points placed across both the floors referenced by "A", "B", "C" and "D".

- Initial plot specifies RSSI values at different coordinates. The signal is strongest near the access point, and intensity drops as we go farther. The intensity is represented by the color, darker the color more the intensity in that area.

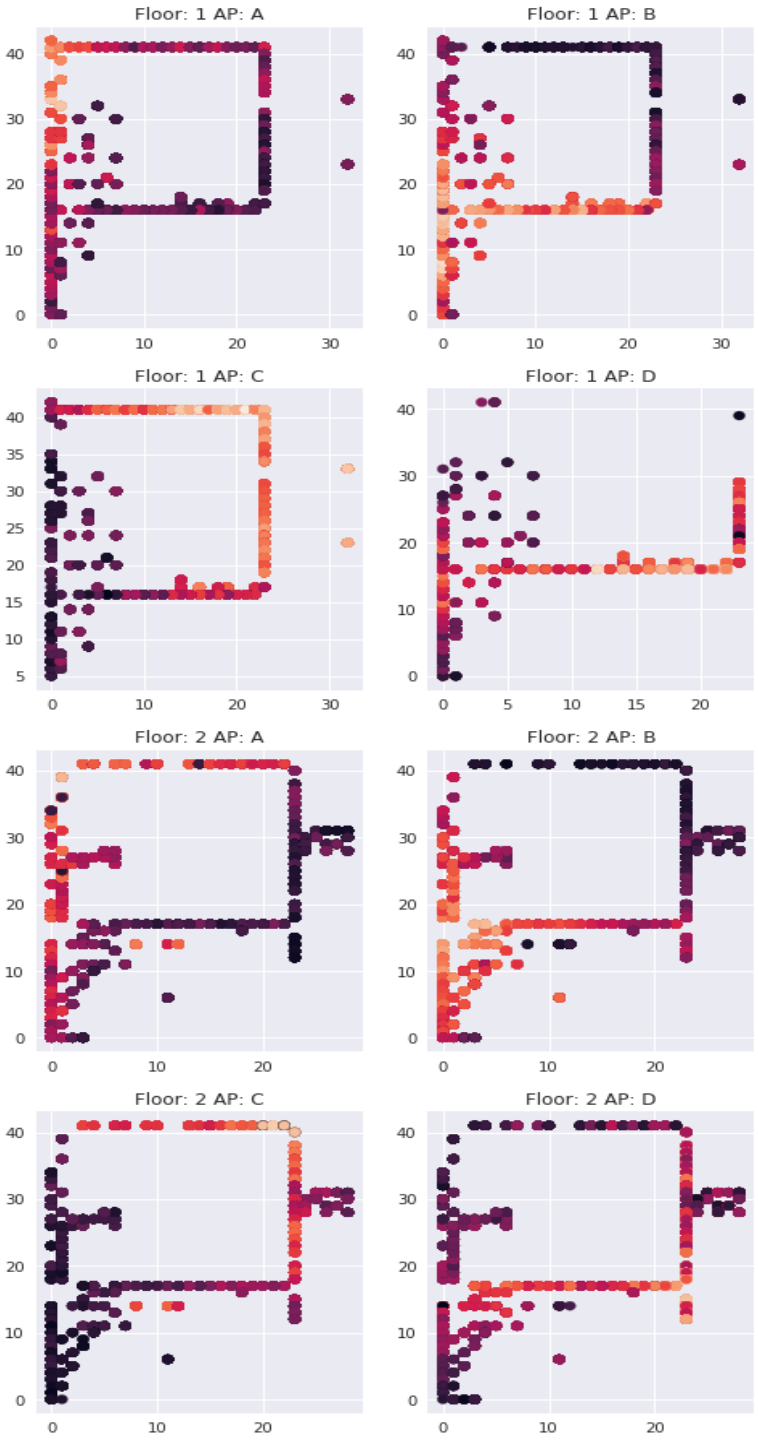


Figure 5-1. Reference dataset fingerprint intensity plot (x-axis-distance in meters, y-axis - distance in meters)

- After plotting the RSSI values, calculation of Euclidean distance of each sampling location is done w.r.t its respective access point.
- A relation graph is plotted between distance to each access point and RSSI (Smaller RSSI values denote stronger signals and therefore correspond to short distances, instability increases as we go farther).

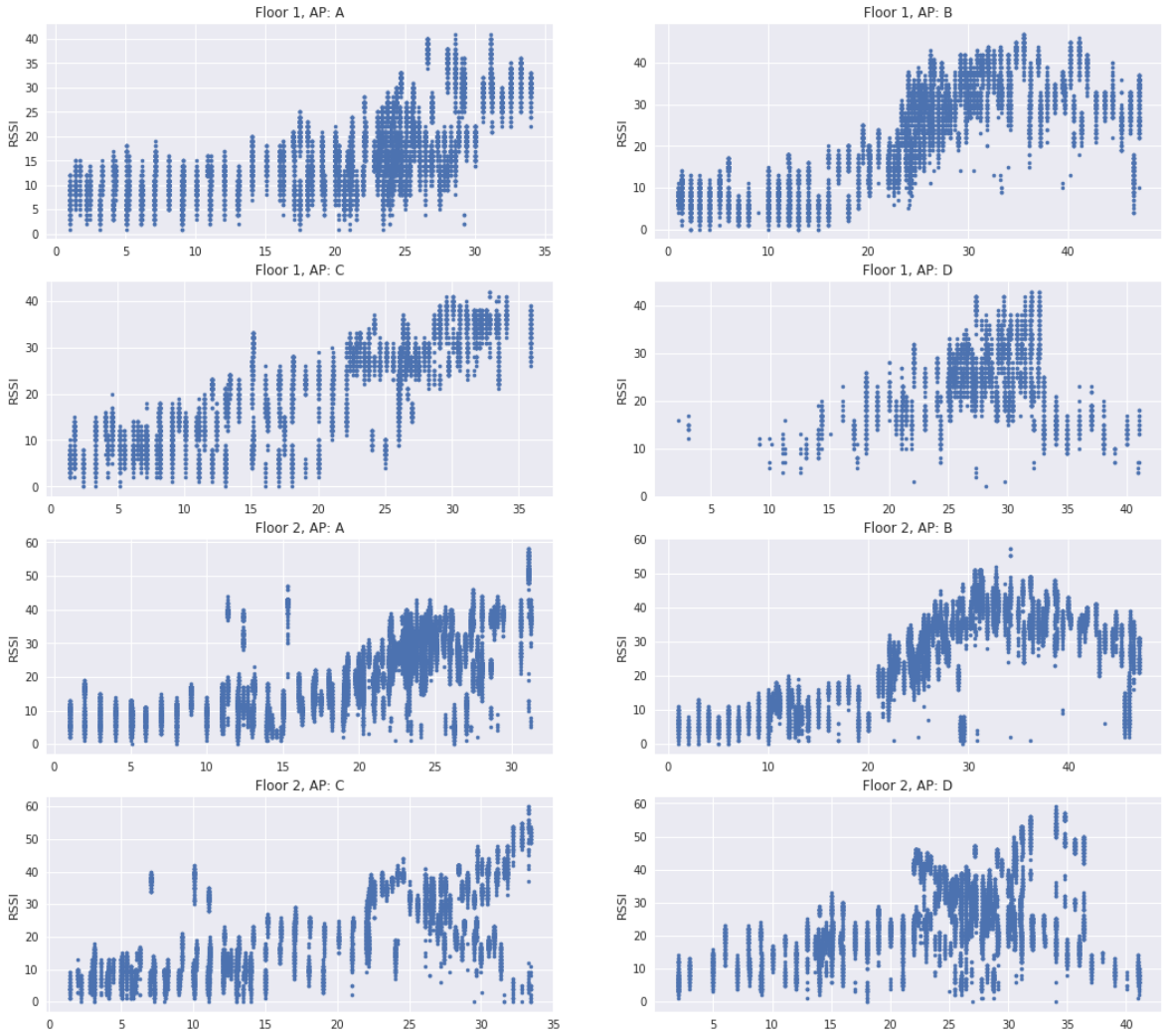


Figure 5-2. Reference dataset fingerprint distance relation plot.(x-axis – distance in meters , y-axis – RSSI values)

- Wild variability of RSSI in almost all locations is a hindrance to accurate location tracking. Following plots denote the use of – min, max, mean and median regressor.

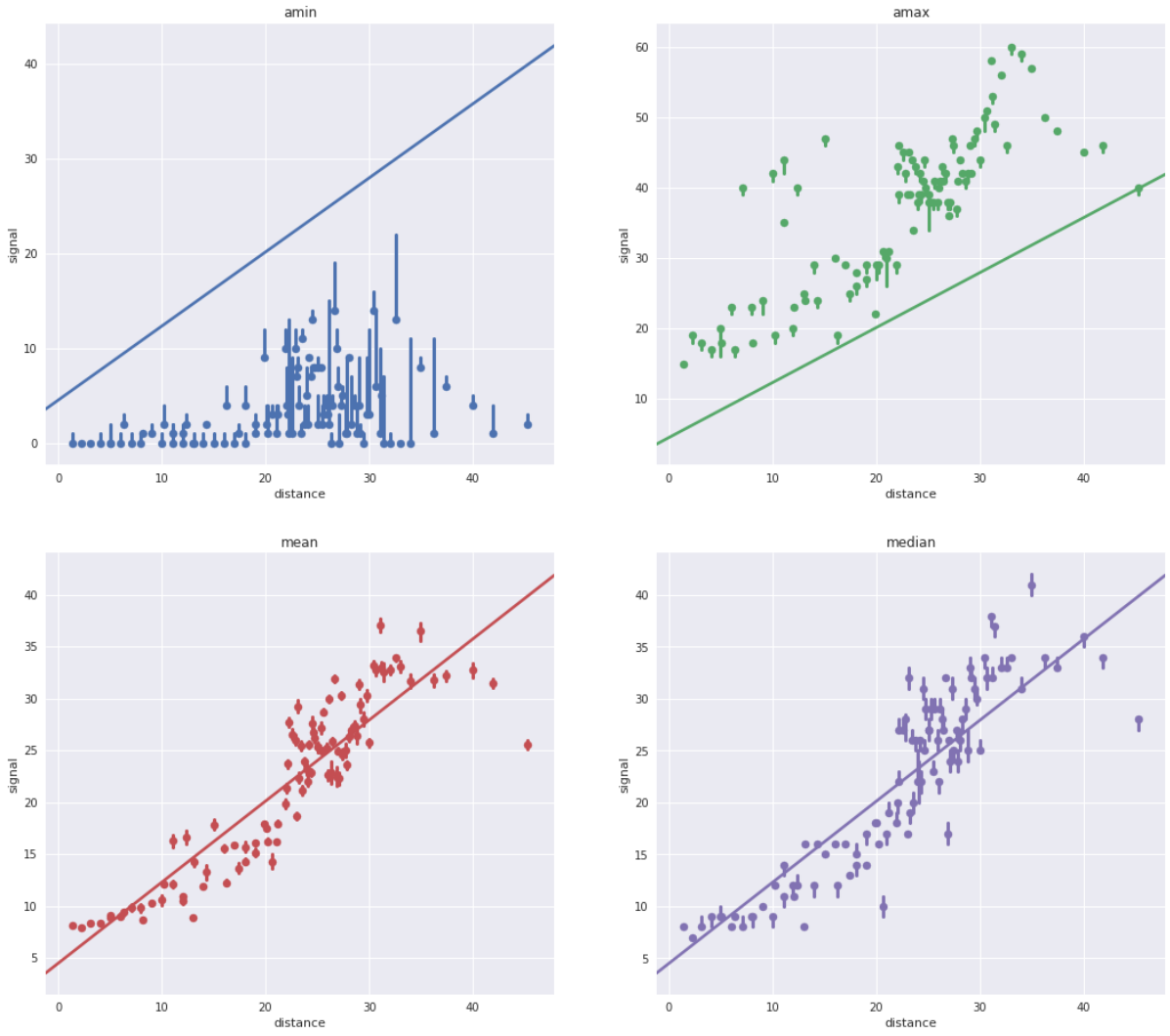


Figure 5-3. Reference dataset distance estimation using min, max, mean, and median regressors. (x-axis -distance in meters, y-axis – distance in meters)

- Plots denote that min and max are not good regressors, whereas mean and median near access points still seem to perform well with distance estimation from access points.
- Non-linearities can be modelled using various other regression algorithms.

5.3.1.1 Preprocessed phase

The next section describes the use of proposed machine learning framework. The data used for modelling has 101973 rows and 6 feature columns. The data which is not used for modelling and used for predictions has 17995 rows with 6 feature columns.

Index	Description	Value
0	session_id	123
1	Target	distance
2	Original Data	(119968, 7)
3	Missing Values	True
4	Numeric Features	4
5	Categorical Features	2
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(83976, 9)
10	Transformed Test Set	(35991, 9)
11	Shuffle Train-Test	True
12	Stratify Train-Test	False
13	Fold Generator	K-Fold
14	Fold Number	10
15	CPU Jobs	-1
16	Use GPU	False
17	Log Experiment	True
18	Experiment Name	diamond1
19	USI	c3dd
20	Imputation Type	simple
21	Iterative Imputation Iteration	None
22	Numeric Imputer	mean
23	Iterative Imputation Numeric Model	None
24	Categorical Imputer	constant
25	Iterative Imputation Categorical Model	None

26	Unknown Categorical Handling	least frequent
27	Normalize	True
28	Normalize Method	z-score
29	Transformation	True
30	Transformation Method	yeo-Johnson
31	PCA	False
32	PCA Method	None
33	PCA Components	None
34	Ignore Low Variance	False
35	Combine Rare Levels	True
36	Rare Level Threshold	0.05
37	Numeric Binning	False
38	Remove Outliers	False
39	Outliers Threshold	None
40	Remove Multicollinearity	True
41	Multicollinearity Threshold	0.95
42	Clustering	False
43	Clustering Iteration	None
44	Polynomial Features	False
45	Polynomial Degree	None
46	Trigonometry Features	False
47	Polynomial Threshold	None
48	Group Features	False
49	Feature Selection	False
50	Feature Selection Method	classic
51	Features Selection Threshold	None
52	Feature Interaction	False
53	Feature Ratio	False
54	Interaction Threshold	None
55	Transform Target	True
56	Transform Target Method	box-cox

Table 5-1. Preprocessing of reference dataset

5.3.1.2 Model training using regression

The next step involves the use of some major supervised algorithms for training our train dataset (data used for modelling). Performance of the supervised learning algorithms is summarized as follows -

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.0033	0.0029	0.0516	1.0000	0.0042	0.0003	3.051
catboost	CatBoost Regressor	0.0926	0.0185	0.1361	0.9998	0.0141	0.0077	7.463
rf	Random Forest Regressor	0.0111	0.0426	0.1955	0.9995	0.0126	0.0008	5.263
dt	Decision Tree Regressor	0.0060	0.0586	0.2246	0.9993	0.0130	0.0005	0.153
knn	K Neighbors Regressor	0.1558	0.0968	0.3111	0.9989	0.0320	0.0146	0.217
xgboost	Extreme Gradient Boosting	0.2232	0.1319	0.3623	0.9985	0.0337	0.0181	27.999
lightgbm	Light Gradient Boosting Machine	0.4988	0.5078	0.7122	0.9942	0.0690	0.0438	0.327
gbr	Gradient Boosting Regressor	1.7911	6.4397	2.5369	0.9262	0.2368	0.1985	2.260
ada	AdaBoost Regressor	4.2449	28.1572	5.3058	0.6775	0.4191	0.4854	0.888
lr	Linear Regression	4.8035	39.9614	6.3213	0.5422	0.4370	0.4948	0.327
br	Bayesian Ridge	4.8034	39.9615	6.3213	0.5422	0.4370	0.4948	0.086
ridge	Ridge Regression	4.8035	39.9614	6.3213	0.5422	0.4370	0.4948	0.076
lar	Least Angle Regression	4.8035	39.9614	6.3213	0.5422	0.4370	0.4948	0.079
huber	Huber Regressor	4.7409	40.5156	6.3649	0.5359	0.4314	0.4776	0.377
omp	Orthogonal Matching Pursuit	5.1924	47.7748	6.9116	0.4528	0.4568	0.5350	0.078
en	Elastic Net	6.7083	75.1958	8.6706	0.1390	0.5685	0.7977	0.080
lasso	Lasso Regression	7.3418	87.3556	9.3455	-0.0003	0.5961	0.8650	0.080
llar	Lasso Least Angle Regression	7.3418	87.3556	9.3455	-0.0003	0.5961	0.8650	0.078

Figure 5-4. Reference Dataset comparison of models on training dataset.

5.3.1.3 Analysis of model comparison

Most of the models generated have an accuracy of 100 percent R-squared which denotes how close data are to the fitted regression line. But R-squared cannot determine whether the coefficient estimates, and predictions are biased, hence evaluation of residual plots is done further (Imam, Kumar and Srivastava, 2018). Any of the top three regression models can further be used for ensemble creation using bagging and boosting methods. For our reference

dataset and for this specific domain, three algorithms seemed to do really well – cat-boost, Random forests and Decision Trees. Errors denotes differences between estimated distance from access points and their actual distances. MAE, MSE, RMSE, RMSLE and MAPE are some common error metrics used for comparison of models.

5.3.1.4 Ensemble of models

A blended model is created which combines all the three models to make predictions. The blended model has been generated with cat-boost, Random forests, decision tress and K-nearest neighbors.

Index	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0346	0.0287	0.1694	0.9997	0.0080	0.0026
1	0.0356	0.0297	0.1723	0.9997	0.0172	0.0032
2	0.0337	0.0088	0.0941	0.9999	0.0057	0.0027
3	0.0358	0.0152	0.1232	0.9998	0.0081	0.0030
4	0.0348	0.0161	0.1268	0.9998	0.0104	0.0031
5	0.0347	0.0319	0.1787	0.9996	0.0183	0.0033
6	0.0350	0.0176	0.1325	0.9998	0.0087	0.0027
7	0.0346	0.0062	0.0785	0.9999	0.0073	0.0027
8	0.0365	0.0117	0.1081	0.9999	0.0093	0.0029
9	0.0383	0.0656	0.2562	0.9992	0.0103	0.0028
Mean	0.0354	0.0231	0.1440	0.9997	0.0103	0.0029
Standard Deviation	0.0012	0.0165	0.0492	0.0002	0.0039	0.0002

Table 5-2.Ensemble model's performance on unseen/test data from Reference dataset.

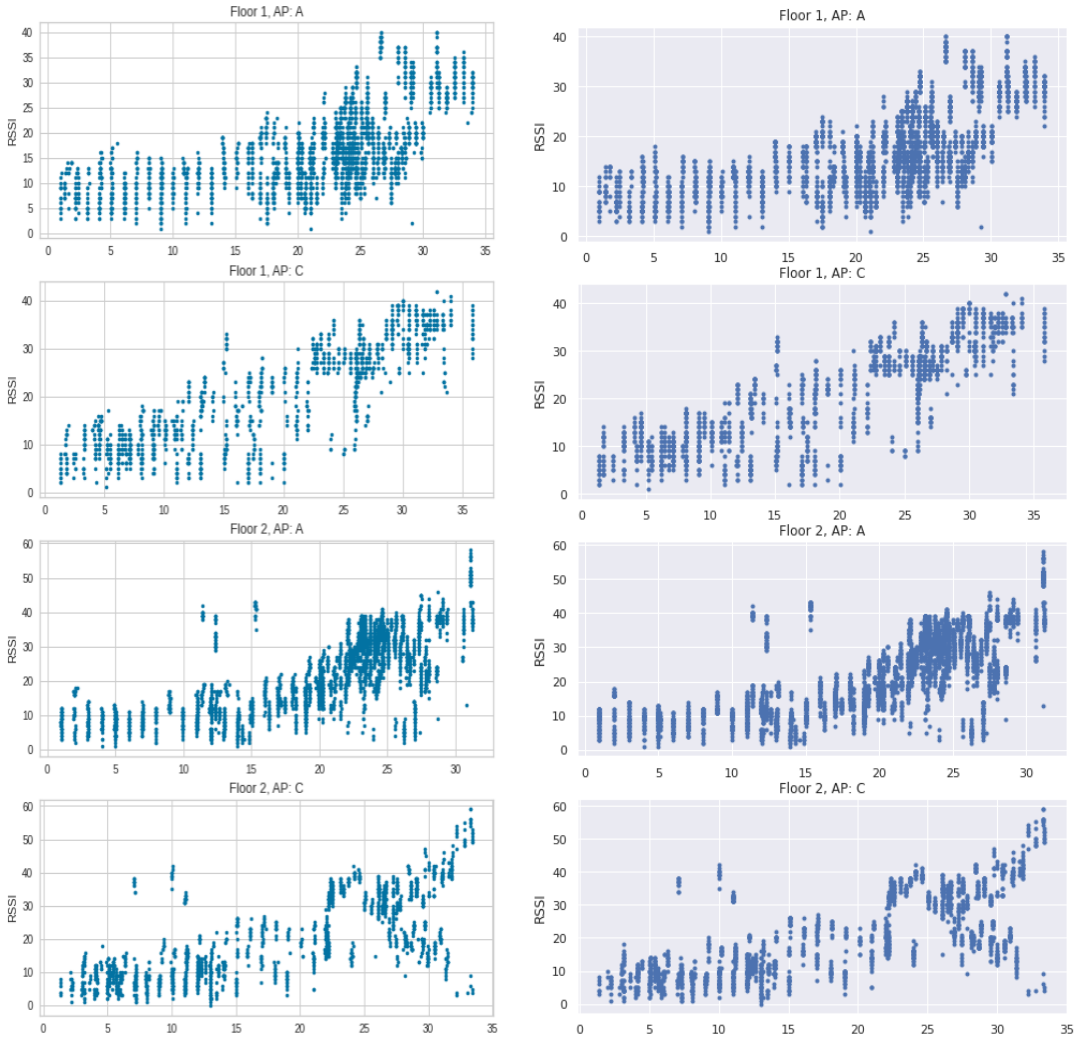
The final predicted model accuracy comes out to 99.9 percent on unseen data, which has not been used for modelling. The blended model can therefore be used as a final model for distance estimation.

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Voting Regressor	0.0314	0.0121	0.1102	0.9999	0.0078	0.0025

Figure 5-5. Final model's prediction performance on unseen/test data.

5.3.1.5 Prediction using Blended Model

The plot denotes a comparison of actual versus predicted distance – RSSI relations.



distance	Label
4.000000	4.062040
26.944387	26.721434
22.135944	22.121305
23.043438	23.060360
21.236761	21.236423

Figure 5-6. Reference Dataset fingerprint actual (left) versus predicted (right) distance relation plot and snapshot of Distance vs Label (bottom) using ensemble of models.

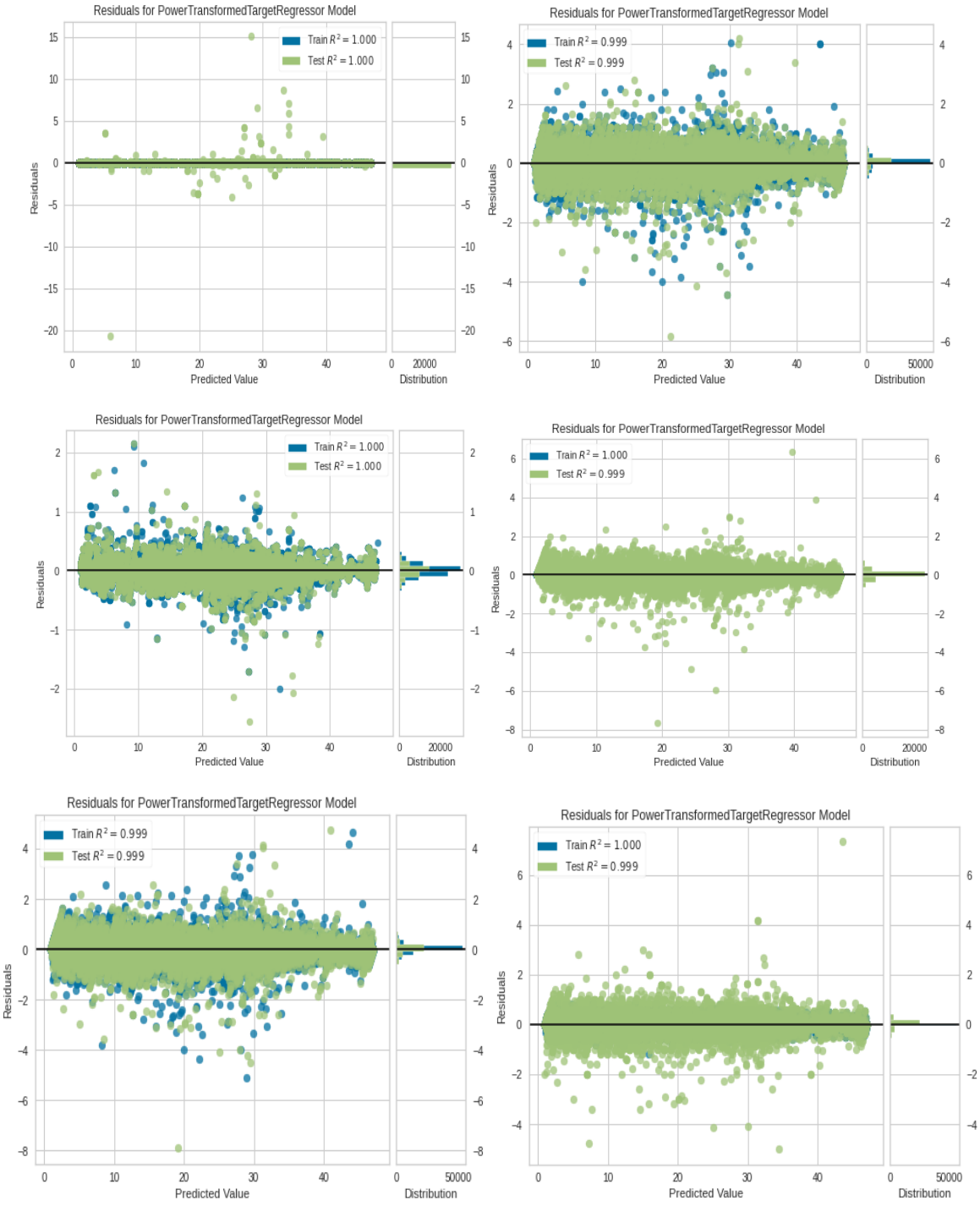
5.3.1.6 Predictions using trained models on test dataset (unseen data for modelling)

Model Name	MAE	MSE	RMSE	R2	RMSLE	MAPE
Decision Tree Regressor	0.0072	0.0572	0.2325	0.9993	0.0121	0.0005
Knn regressor	0.1722	0.1113	0.3336	0.9987	0.0342	0.0160
Catboost regressor	0.0951	0.0197	0.1402	0.9998	0.0142	0.0078
Tuned_knn regressor	0.1572	0.0830	0.2878	0.9990	0.0308	0.0152
Bagged_knn regressor	0.1677	0.0968	0.3111	0.9989	0.0317	0.0155
Boosted_knn regressor	0.1212	0.0640	0.2529	0.9993	0.0270	0.0114
Blender	0.0354	0.0231	0.1440	0.9997	0.0103	0.0029
Stacker	0.0321	0.0046	0.0665	0.9999	0.0058	0.0023

Table 5-3.Trained models' performance on test set of reference dataset

This plot summarizes how after combining different models and tuning various hyperparameters, the errors are brought down, and accuracy is increased at the same time. As evident from the tabular plot, the blender and stacker which are ensemble of models seems to outperform them all. The distance estimation along with the reduced errors using the ensemble of models seem to be an efficient framework over the mean and median regressors which might not be able to include interferences and other non-linearities in the environment.

5.3.1.7 Residual Plots on test data



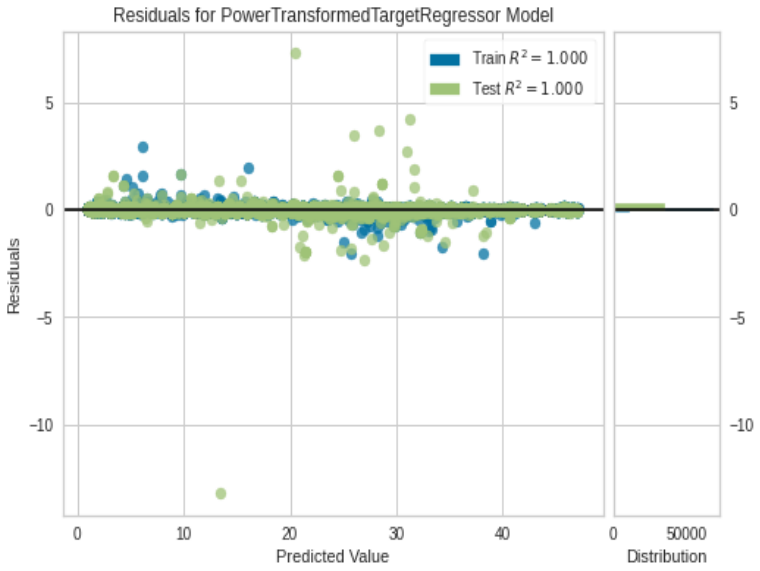
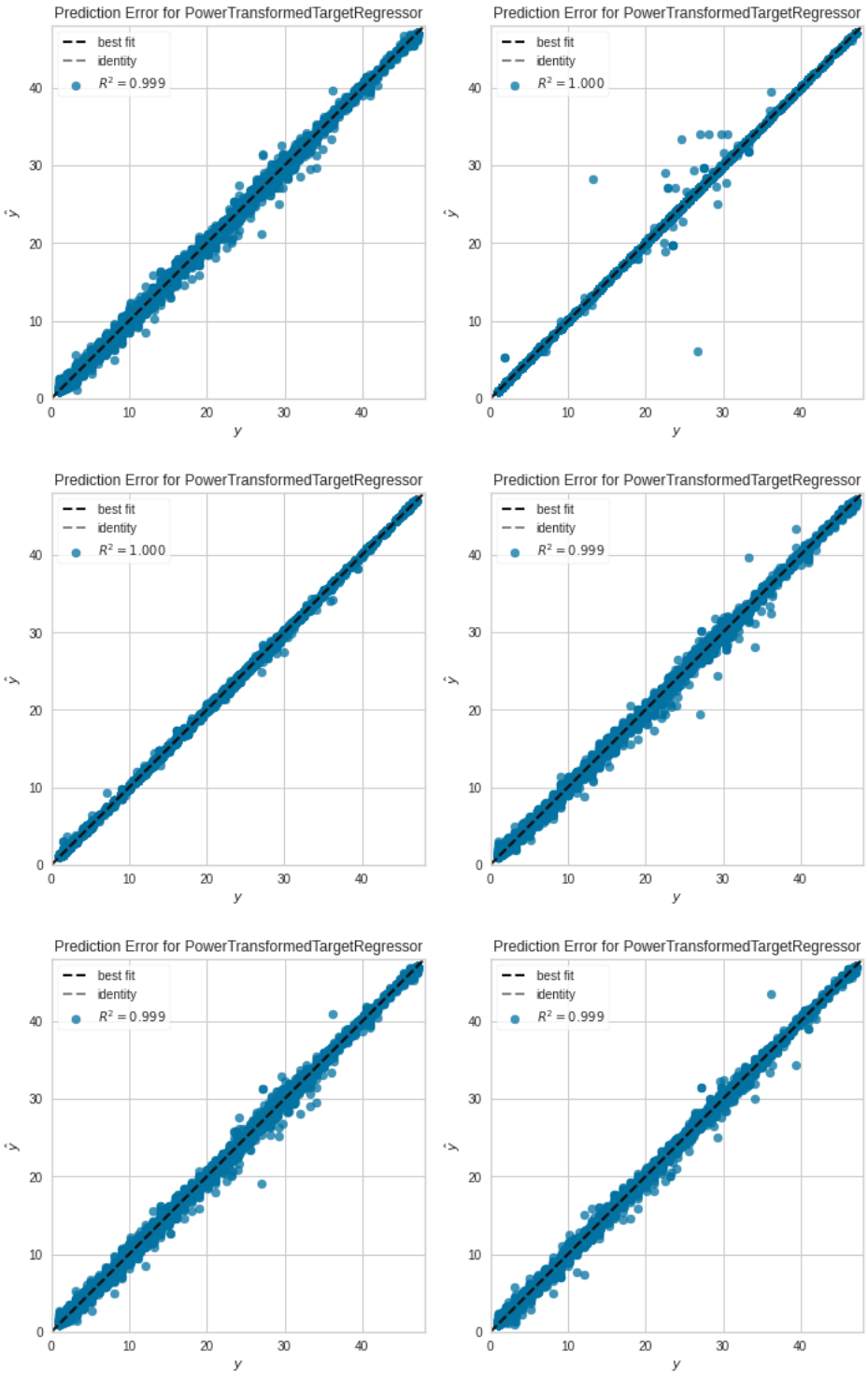


Figure 5-7.Residual plots for all trained model (Decision Trees, K-nearest neighbors, Cat-boost, Tuned K-nearest neighbors, Bagged Knns, Boosted Knns and Blender model) on unseen/test data.

Residual plots follow property that all errors are independent and normally distributed. The normal distribution is clear from the distribution plot besides the residual plots in the above figure. Few characteristics of a good plot are high-density points close to origin and low density of points away from origin (Gohar, 2020). A generalized model depicts symmetricity along its origin. The plots follow order – Decision Trees, K-nearest neighbors, Cat-boost, Tuned K-nearest neighbors and Ensemble of models (bagged and boost). The ensemble models have highest density of points close to origin. The final model is combined with decision trees, Cat-boost and K-nearest neighbors. This model has minimum outliers compared to any other model rendering it as best model to be used for deployment.

5.3.1.8 Prediction Error plots on test data



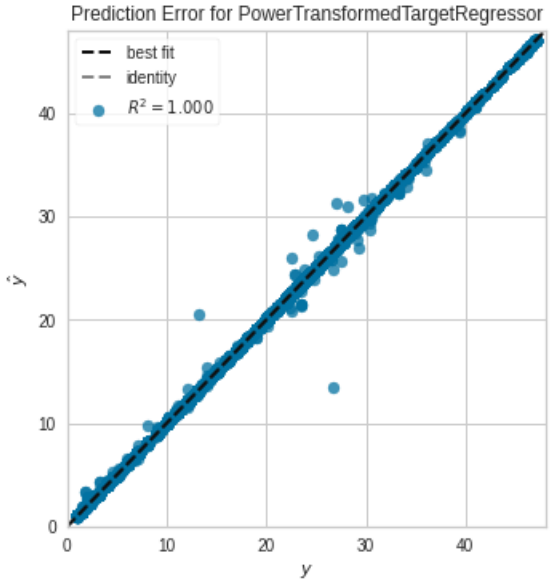


Figure 5-8. Prediction error plots for all trained (Follows order from left to right - Decision Trees, K-nearest neighbors, Cat-boost, Tuned K-nearest neighbors, Bagged Knns, Boosted Knns and Blender model).

The plots follow order from left to right – Decision Trees, K-nearest neighbors, Cat-boost, Tuned K-nearest neighbors, and Ensemble of models (bagged and boost). Most of them seems to be less biased with low variance. However, K-nearest neighbors seem to have some outliers which represents some of the prediction errors, hence demonstrating that even after modelling, there always seems to be some features which are not weighed in and hence can affect the prediction on unseen data. Most of the fingerprinting research is concerned with the use of K-nearest neighbors for distance estimation or location determination. This graph makes it obvious on how even after tuning K-nearest neighbors, significant portion of outliers seem to be persistent in the final predictions.

5.3.1.9 Interpretation of models

The interpretation of model is one very useful tool supplied with Pycaret which has been integrated with our framework to explain the predictions of the model. The model is interpreted using SHAP (Shapley Additive Explanations) (Lundberg and Lee, 2017). Accuracy versus interpretability trade-off has always been relevant for model success. The use of SHAP simplifies this issue. SHAP identifies the new class of additive feature importance measures. The feature is currently designed to accommodate only tree-based model. The interpretation has been done for decision tree regression and cat-boost regression.

Figure 5-9. Interpretation of cat-boost regression model on unseen data/test set.

.

The first plot shows the interpretation of the cat-boost regression. It depicts how x and y grid coordinates are very important for the prediction. Access point B seems to be a very good candidate for fingerprint predictions.

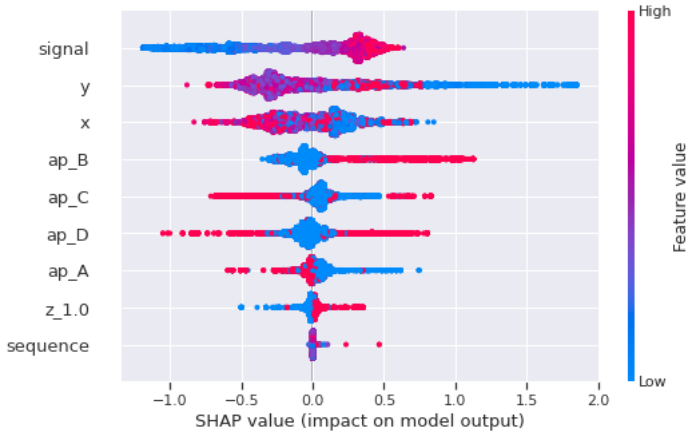


Figure 5-10. Interpretation of decision trees regression model on unseen data/test set.

The second plot explains the decision tree model. The interpretation follows the cat-boost regression's interpretation. There are some minor differences which varies from domain to interferences involved.

5.3.2 Distance Estimation using transformed dataset generated via Dataset Generation Phase

This section covers all the regression phase for distance estimation using the transformed dataset. The dataset is transformed initially with the dataset generation phase where Minkowski's distance measure is used for feature transformation using the latitudes and longitudes of the locations. This dataset uses 2048 datapoints or instances. Split has been made as 85% data for modelling (training set) and 15% data for testing out on trained model. The regression phase process for this dataset is similar to the previous phase. This section covers only final comparisons on test datasets.

5.3.2.1 Preprocessed Phase

The preprocessed phase covers the necessary transformation required for the dataset to be used further for model generation. Below table covers a brief overview on some of the preprocessed features. The training test set is again divided into training and validation dataset for avoiding overfitting.

Index	Description	Value
0	session_id	123
1	Target	distance
2	Original Data	(1334, 14)
3	Missing Values	False
4	Numeric Features	3
5	Categorical Features	9
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(933, 32)
10	Transformed Test Set	(401, 32)
11	Shuffle Train-Test	True
12	Stratify Train-Test	False
13	Fold Generator	K-Fold
14	Fold Number	10

15	CPU Jobs	-1
16	Use GPU	True
17	Log Experiment	False
18	Experiment Name	Distance-estimation
19	USI	89fb
20	Imputation Type	iterative
21	Iterative Imputation Iteration	20
22	Numeric Imputer	mean
23	Iterative Imputation Numeric Model	Light Gradient Boosting Machine
24	Categorical Imputer	constant
25	Iterative Imputation Categorical Model	Light Gradient Boosting Machine
26	Unknown Categorical Handling	Least frequent
27	Normalize	True
28	Normalize Method	z-score
29	Transformation	False
30	Transformation Method	None
31	PCA	False
32	PCA Method	None
33	PCA Components	None
34	Ignore Low Variance	False
35	Combine Rare Levels	False
36	Rare Level Threshold	None
37	Numeric Binning	False
38	Remove Outliers	False
39	Outliers Threshold	None
40	Remove Multicollinearity	True
41	Multicollinearity Threshold	0.850000
42	Clustering	False
43	Clustering Iteration	None
44	Polynomial Features	False
45	Polynomial Degree	None

46	Trigonometry Features	False
47	Polynomial Threshold	None
48	Group Features	False
49	Feature Selection	False
50	Feature Selection Method	classic
51	Features Selection Threshold	None
52	Feature Interaction	False
53	Feature Ratio	False
54	Interaction Threshold	None
55	Transform Target	False
56	Transform Target Method	box-cox

Table 5-4. Preprocessing of Transformed Location dataset (generated using dataset generation)

5.3.2.2 Predictions using the trained models on test dataset

The trained models are finally tested out using the test set or the unseen data which is not used for modelling. Below is a brief overview on the R-squared accuracy on the test sets. The cat-boost regression and blended model seems to outperform the other models. However, as mentioned before R-squared accuracy alone isn't enough to cover all the interpretations of the model. The blended model and Stacking model achieve 99 % accuracy with low error counts.

Model Name	**R2**
Cat-Boost Regressor	1.0
K Neighbors Regressor	0.9956
Decision Tree Regressor	0.9972
Blended model	0.9991
Stacking model	1.0

Table 5-5.Trained models' performance on transformed Location dataset (generated using dataset generation)

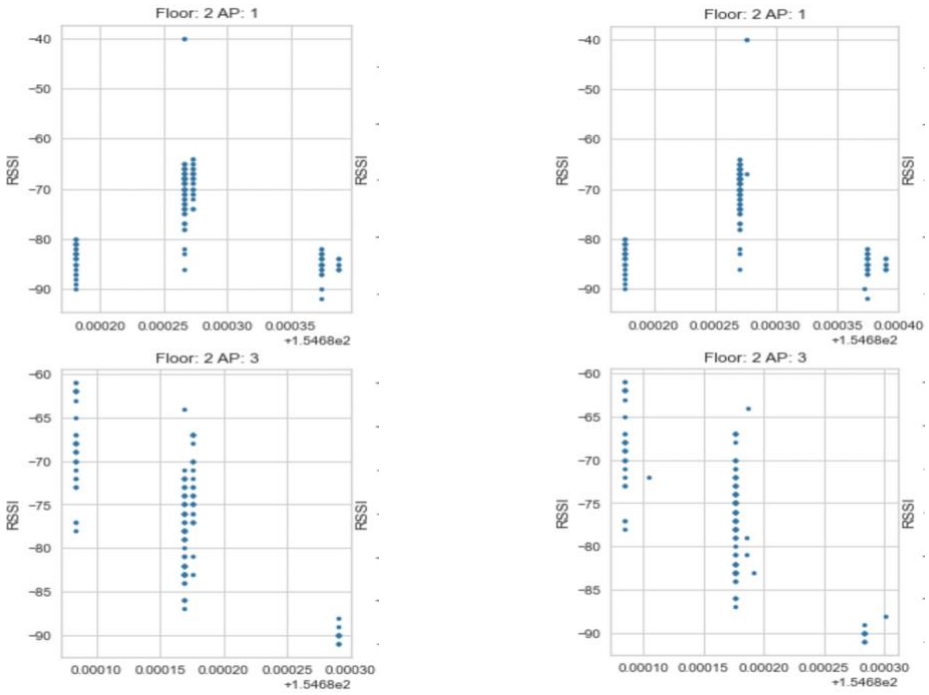
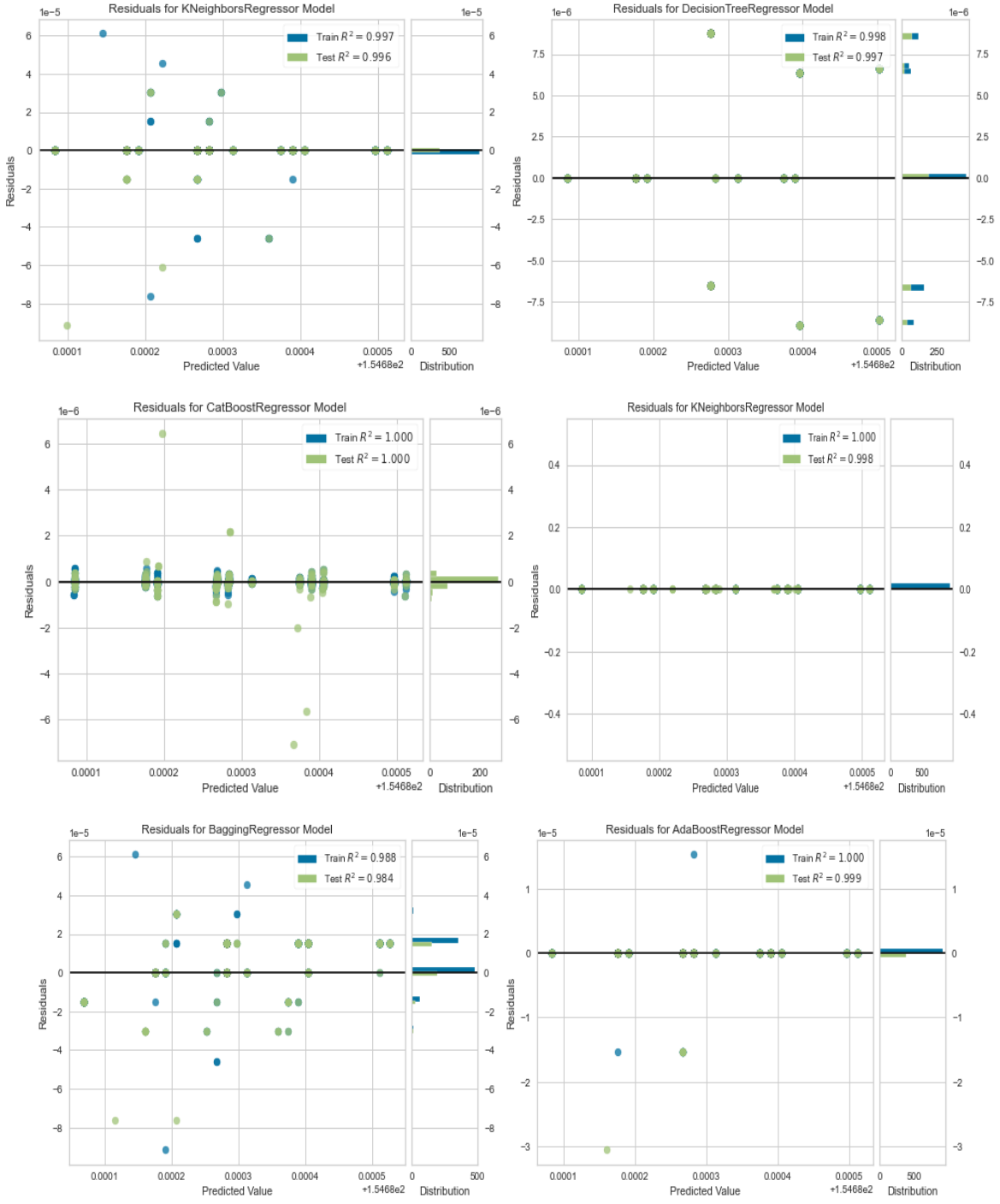


Figure 5-11. Final model's predicted distance versus RSSI plot on unseen/test data of transformed location dataset. (x-axis – distance in meters, y-axis – RSSI values in dB).

5.3.2.3 Residual Plots

The residual plot follows order K-nearest neighbors, Decision trees regression, Cat-boost regression, Tuned K-nearest neighbors and Ensemble of models. Decision trees seem to perform well without significant outliers. Cat-boost performs slightly better than K-nearest neighbors which has significant outliers. However, tuned K-nearest neighbors seem to perform better than Cat-boost and K-nearest neighbors. Ensemble of models are used at the end for making final predictions on the unseen datapoint which haven't been used for modelling.


Residuals for KNeighborsRegressor Model
Train $R^2 = 0.997$
Test $R^2 = 0.996$
Residuals for DecisionTreeRegressor Model
Train $R^2 = 0.998$
Test $R^2 = 0.997$
Residuals for CatBoostRegressor Model
Train $R^2 = 1.000$
Test $R^2 = 1.000$
Residuals for KNeighborsRegressor Model
Train $R^2 = 1.000$
Test $R^2 = 0.998$
Residuals for BaggingRegressor Model
Train $R^2 = 0.988$
Test $R^2 = 0.984$
Residuals for AdaBoostRegressor Model
Train $R^2 = 1.000$
Test $R^2 = 0.999$
Residuals
Predicted Value
Distribution

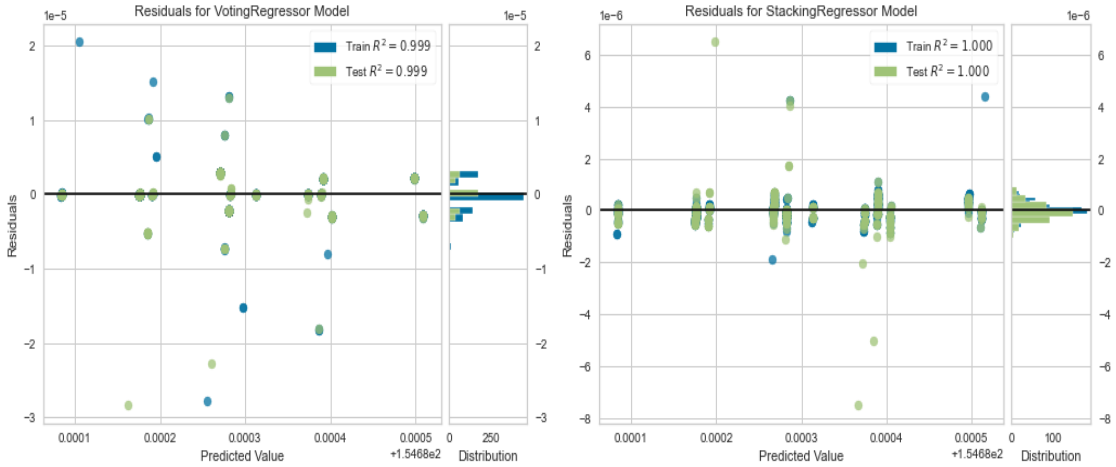


Figure 5-12. Residual plots for all trained model (Decision Trees, K-nearest neighbors, Cat-boost, Tuned K-nearest neighbors, Bagged Knns, Boosted Knns, Blender and Stacking model) on unseen/test data (x-axis – distance in meters, y-axis – Residual values).

5.3.2.4 Interpretation of model

Cat-boost and decision trees have been used for interpretation of model. For Cat-boost, access point 3 and access point 4 (the sensors) are major contributors towards distance estimations. The locations of wireless sensor lab and exit lobby from the floor plan have significant fractions towards distance estimation, denoting the fact that those two locations are appropriate for data collection. Decision trees otherwise reflect the same conclusion as explained in the following plots.

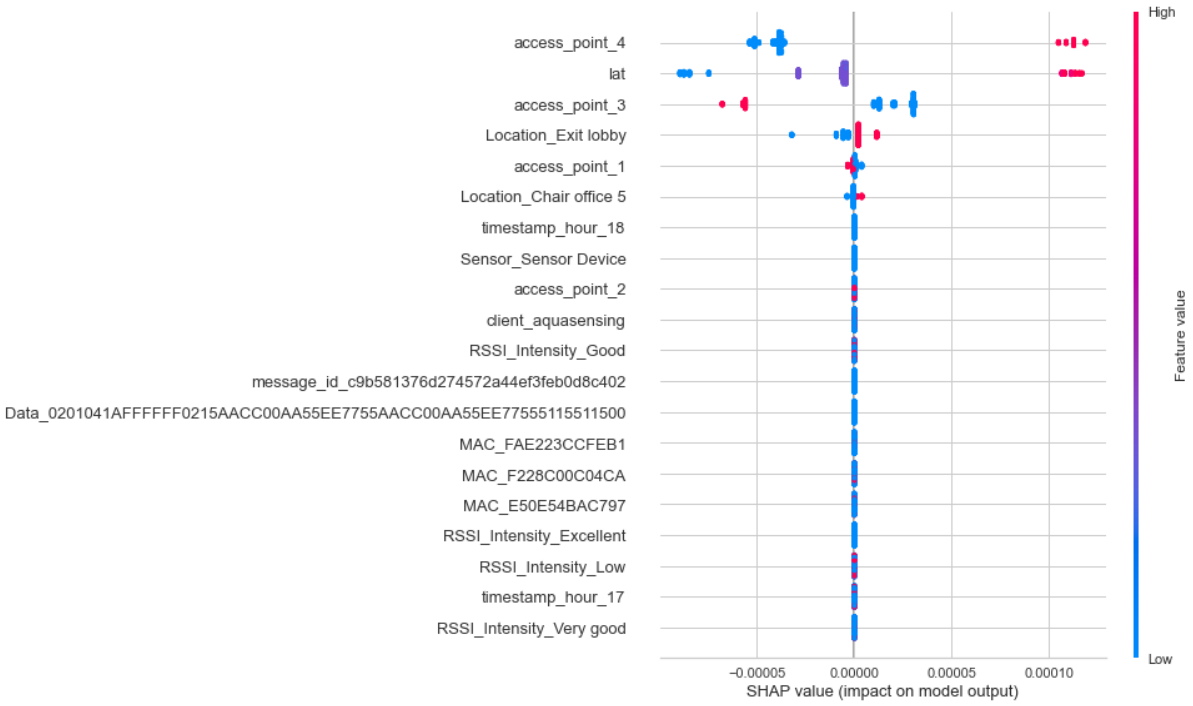


Figure 5-13. Interpretation of cat-boost regression model on unseen data/test set of transformed location dataset.

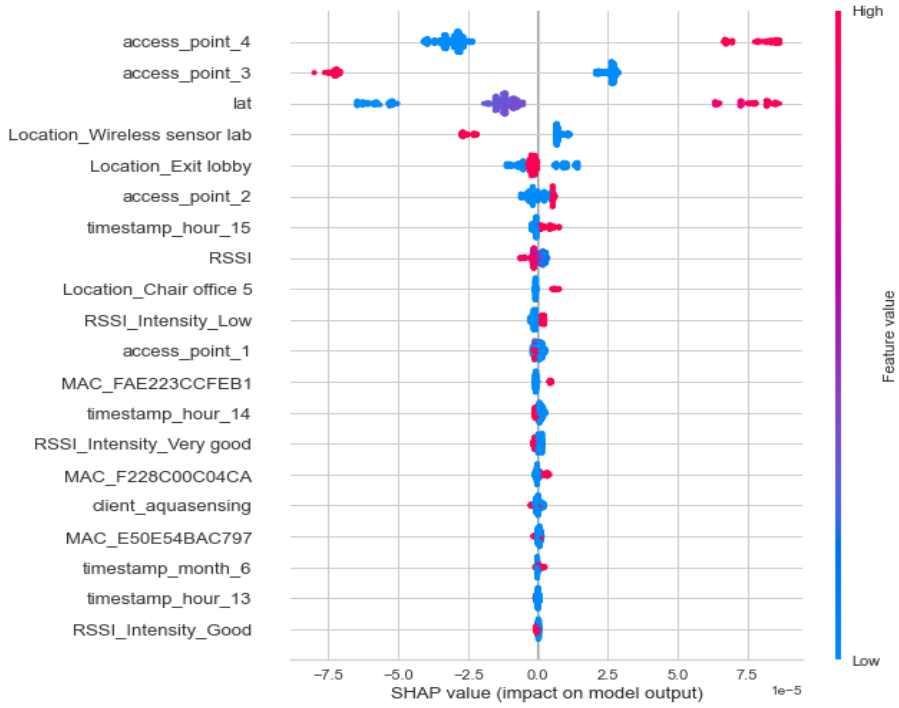


Figure 5-14. Interpretation of Decision Tree regression model on unseen data/test set of transformed location dataset.

5.4 Classification Phase

This section describes a classification pipeline designed for fingerprint intensity classification. The fingerprint used in our use case is RSSI, and the dataset used for training our classifier has been generated using the dataset generation phase. The classification is a multi-label classification. The transformed dataset has multiple classes for RSSI intensity which is our target feature. This problem is therefore a multi-label classification machine learning use case.

5.4.1 Preprocessed Phase

The preprocessed phase evaluates the transformed dataset and performs necessary preprocessing steps. The target feature is hot encoded and dataset imbalance is resolved using SMOTE (Synthetic Minority Oversampling Technique) which synthesizes new examples from existing samples. The data augmentation improves performance of the trained model. A minority class instance of RSSI intensity is selected (i.e., Excellent/Outstanding RSSI Intensity sample values which are less in number). The model generation phase relies on the transformed dataset generated using the dataset generation phase. The dataset consists of 2052 data instances with 14 feature columns. It is split into 70 percent training set and 30 percent test set. The training instances are used for training the classifier or model and rest of instances are used to test out the trained model.

Index	Description	Value
0	session_id	111
1	Target	RSSI_Intensity
2	Target Type	Multiclass
3	Label Encoded	Excellent: 0, Good: 1, Low: 2, Outstanding: 3, Very good: 4, Very low: 5
4	Original Data	(2052, 14)
5	Missing Values	False
6	Numeric Features	4
7	Categorical Features	8
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(1436, 1456)
12	Transformed Test Set	(616, 1456)
13	Shuffle Train-Test	True

14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	True
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	9ba5
22	Imputation Type	iterative
23	Iterative Imputation Iteration	10
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	Light Gradient Boosting Machine
26	Categorical Imputer	mode
27	Iterative Imputation Categorical Model	Light Gradient Boosting Machine
28	Unknown Categorical Handling	least_frequent
29	Normalize	True
30	Normalize Method	z-score
31	Transformation	True
32	Transformation Method	quantile
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	True
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	False
43	Multicollinearity Threshold	None
44	Clustering	False
45	Clustering Iteration	None
46	Polynomial Features	False
47	Polynomial Degree	None
48	Trignometry Features	False
49	Polynomial Threshold	None
50	Group Features	False

51	Feature Selection	False
52	Feature Selection Method	classic
53	Features Selection Threshold	None
54	Feature Interaction	False
55	Feature Ratio	False
56	Interaction Threshold	None
57	Fix Imbalance	True
58	Fix Imbalance Method	SMOTE

Table 5-6.Preprocessing of transformed location dataset for classification phase.

5.4.2 Comparison of models on training dataset

Training of the models is done using various supervised classification algorithms. The tree algorithms seem to perform better than other algorithms. Similar to regression phase, Accuracy is not the only metric to be relied on. Area under Receiver operating characteristic curve (AUC), precision, recall, F1-score etc. are some of the other metrics that can be used for evaluation of trained models.

Index	Model	Accuracy	AUC	Recall	Precision	F1	Kappa	MCC	TT (Sec)
catboost	Cat-Boost Classifier	1.0000	0.4000	1.0000	1.0000	1.0000	1.0000	1.0000	6.6930
dt	Decision Tree Classifier	0.9989	0.4000	0.9791	1.0000	0.9994	0.9984	0.9984	0.0240
gbc	Gradient Boosting Classifier	0.9989	0.4000	0.9791	1.0000	0.9994	0.9984	0.9984	2.6780
lightgbm	Light Gradient Boosting Machine	0.9989	0.4000	0.9791	1.0000	0.9994	0.9984	0.9984	0.1700
rf	Random Forest Classifier	0.9593	0.3991	0.8351	0.9548	0.9550	0.9398	0.9410	0.7380

lr	Logistic Regression	0.9368	0.3964	0.8025	0.9305	0.9321	0.9063	0.9071	0.5690
et	Extra Trees Classifier	0.9325	0.3969	0.8244	0.9278	0.9286	0.9002	0.9013	0.7720
knn	K Neighbors Classifier	0.9250	0.3949	0.8591	0.9236	0.9228	0.8893	0.8904	0.4790
lda	Linear Discriminant Analysis	0.8145	0.3836	0.6034	0.8187	0.8060	0.7206	0.7293	0.3990
ada	Ada Boost Classifier	0.7567	0.3467	0.6867	0.6107	0.6654	0.6244	0.6936	0.1800
ridge	Ridge Classifier	0.7492	0.0000	0.5567	0.7239	0.7332	0.6265	0.6299	0.0350
svm	SVM - Linear Kernel	0.7385	0.0000	0.6204	0.7407	0.7279	0.6124	0.6222	0.1240
nb	Naive Bayes	0.2165	0.2342	0.4510	0.2139	0.1366	0.0793	0.1109	0.0250
qda	Quadratic Discriminant Analysis	0.1038	0.1242	0.0987	0.1041	0.0880	0.0492	0.0618	0.2620

Table 5-7.Comparison of classifiers on training data of transformed location dataset.

5.4.3 Predictions on data not used for modelling (Test dataset)

After comparison of classifier models, models are created using logistic regression, light gradient boosting, cat-boost, k-nearest neighbors and random forests. The trained models are then evaluated on the test dataset. The models are then tuned to optimize AUC. The tuned versions have better generalization and less misclassified classes. An ensemble of models is generated using Bagging and Boosting which outperforms the tuned versions of classifiers. A blender model composed of all the above-mentioned classifiers (light gradient boosting, logistic regression, k-nearest neighbors and random forests) is generated and evaluated on test set. The following table depicts the performance of all with other evaluation metrics and coefficients.

Index	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Logistic Regression	0.9451	0.9907	0.7011	0.9439	0.9424	0.9201	0.9205
1	Light Gradient Boosting classifier	0.9975	1.0	0.9444	0.9975	0.9973	0.9964	0.9964
2	Cat-boost Classifier	0.9975	0.9999	0.9444	0.9975	0.9973	0.9964	0.9964
3	K-nearest neighbors	0.9327	0.9981	0.7032	0.9319	0.9309	0.9023	0.9026
4	Random forests	0.9751	0.9979	0.7251	0.9730	0.9729	0.9638	0.9640
5	Tuned Logistic Regression	0.9526	0.9936	0.7396	0.9511	0.9513	0.9314	0.9317
6	Tuned light gradient boosting classifier	0.9950	1.0	0.7778	0.9925	0.9938	0.9928	0.9928
7	Tuned cat-boost classifier	0.9975	0.9992	0.9444	0.9975	0.9973	0.9964	0.9964
8	Tuned K-nearest neighbors	0.9626	0.9902	0.8849	0.9639	0.9615	0.9457	0.9462

9	Tuned random forests	0.9975	1.0	0.9444	0.9975	0.9973	0.9964	0.9964
10	Bagged logistic regression	0.9501	0.9920	0.7349	0.9487	0.9489	0.9277	0.9281
11	Bagged light gradient boosting classifier	0.9950	1.00	0.7778	0.9925	0.9938	0.9928	0.9928
12	Bagged cat-boost classifier	0.9975	1.0	0.9444	0.9975	0.9973	0.9964	0.9964
13	Bagged K-nearest neighbor	0.9426	0.9899	0.8731	0.9437	0.9417	0.9167	0.9169
14	Bagged random forests	0.9975	1.0	0.9444	0.9975	0.9973	0.9964	0.9964
15	Boosted logistic regression	0.9501	0.9920	0.7379	0.9487	0.9489	0.9277	0.9281
16	Boosted light gradient boosting classifier	0.3616	0	0.1667	0.1308	0.1921	0	0

17	Boosted Cat-boost classifier	0.9975	0.9992	0.9444	0.9975	0.9973	0.9964	0.9964
18	Boosted random forests	0.9975	1.0	0.9444	0.9975	0.9973	0.9964	0.9964
19	Blender	0.9925	0.9989	0.9248	0.9926	0.9921	0.9892	0.9892
20	Stacker	0.9975	0.9993	0.9444	0.9975	0.9973	0.9964	0.9964

Table 5-8. Training models' performance on unseen/test data of transformed location dataset

5.4.4 Area Under ROC curve

The Receiver Operator Characteristic curve (ROC) has many faces when being analyzed. It's important to understand the interpretation behind each of the ROC curve. The ROC analysis isn't only about the cost-sensitive learning. The classifiers on the convex hull achieve the best accuracy for class distributions whereas the classifiers lying below the convex hull are sub-optimal(*ICML'04 tutorial on ROC analysis*, no date). Before analyzing the classifiers on the final predictions, here's a deep dive into the curves.

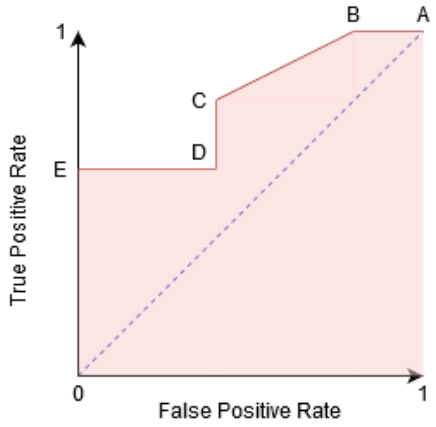


Figure 5-15. AUC explanation with points on the curve.

Specificity is lowest and sensitivity is highest at point A providing evidence on correct classification of all positive class instances and incorrect classification of all negative class instances. As the curve shifts to point B, incorrect classification of negative class instances starts decreasing

denoting that threshold at this point is better than point A. The specificity stays same at C and D points but as noticeable from the reference plot, point C has higher sensitivity than point D. The number of correctly classified positive instances increase for same value of specificity and lastly, point E has highest specificity and hence highest number of correct classifications for negative class points. The analysis provided for the model generated on test set follows the order – Logistic regression, Cat-boost, K-nearest neighbors, Light-gradient boosting, random forests with their tuned counterparts and ensemble of models with bagged, boosted and blender models. Each plot has confusion matrix besides it showing the number of predictions made using test set.

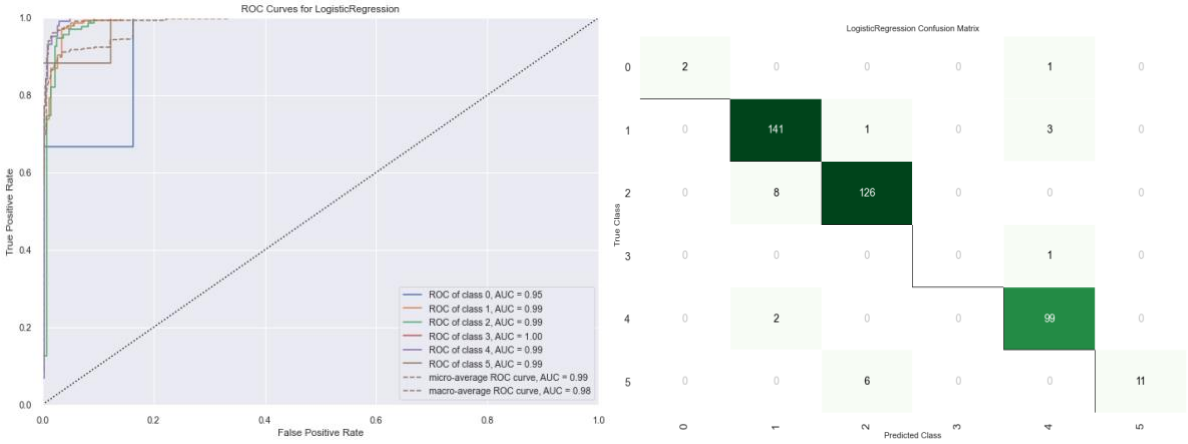


Figure 5-16. AUC and confusion matrix of Logistic regression on unseen/test data of transformed location dataset.

The logistic regression being a rather simple implementation faces issues in classifying all instances correctly. The prediction miss-outs or misclassification of labels is quite significant for classes ‘1’,’2’,’4’ which represent ‘Good’, ‘Low’, ‘Very good’ RSSI intensity levels.

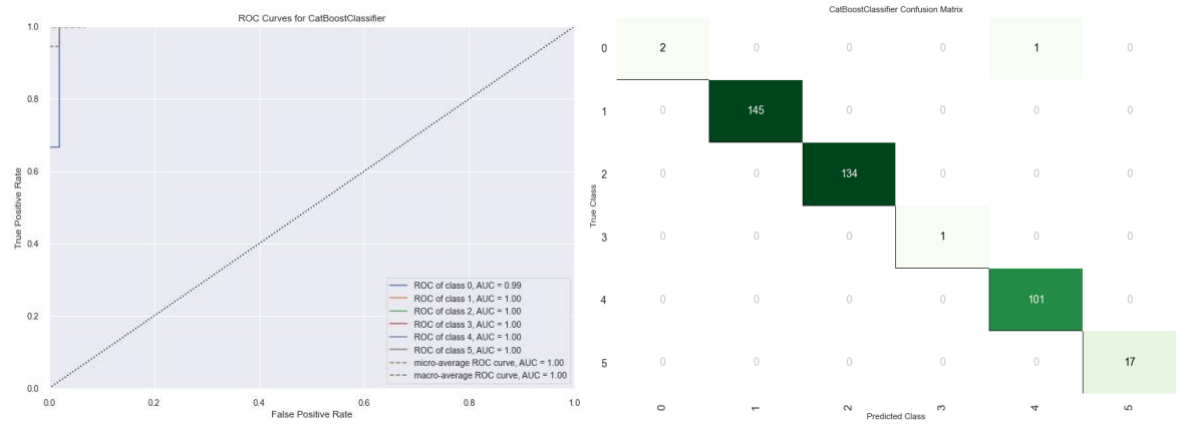


Figure 5-17. AUC and confusion matrix of Cat-boost classifier on unseen/test data of transformed location dataset.

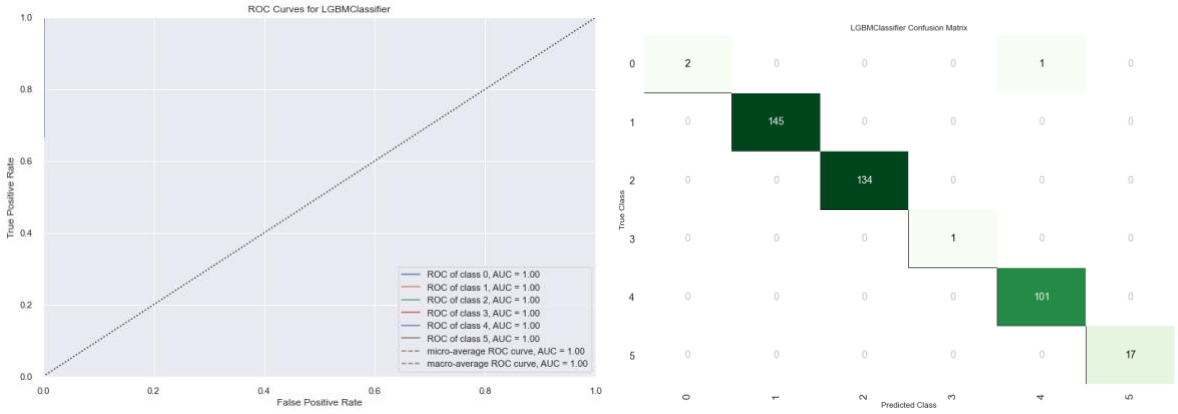


Figure 5-18. AUC and confusion matrix of Light gradient boosting classifier on unseen/test data of transformed location dataset.

Both light gradient boosting, and Cat-boost proved to be good classifiers, reducing the misclassifications of the labels and maintaining accuracy of the predictions.

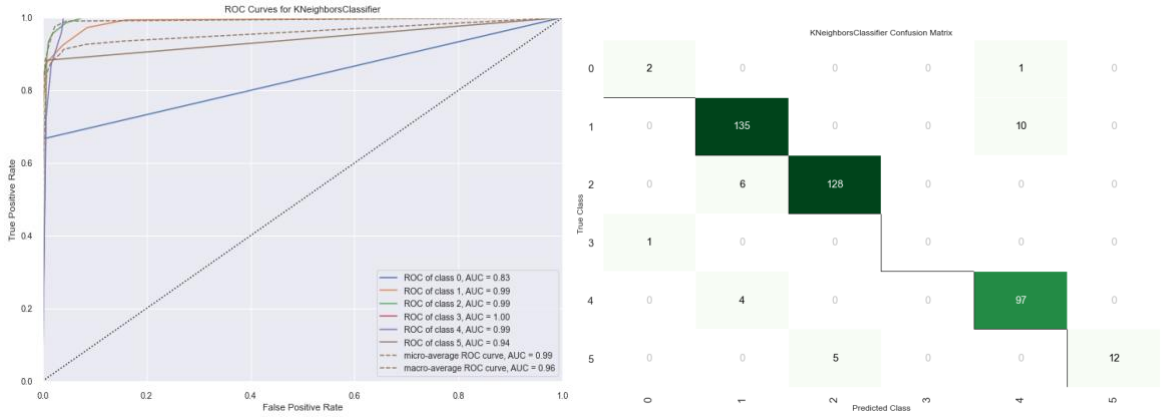


Figure 5-19. AUC and confusion matrix of K-nearest neighbors on unseen/test data of transformed location dataset.

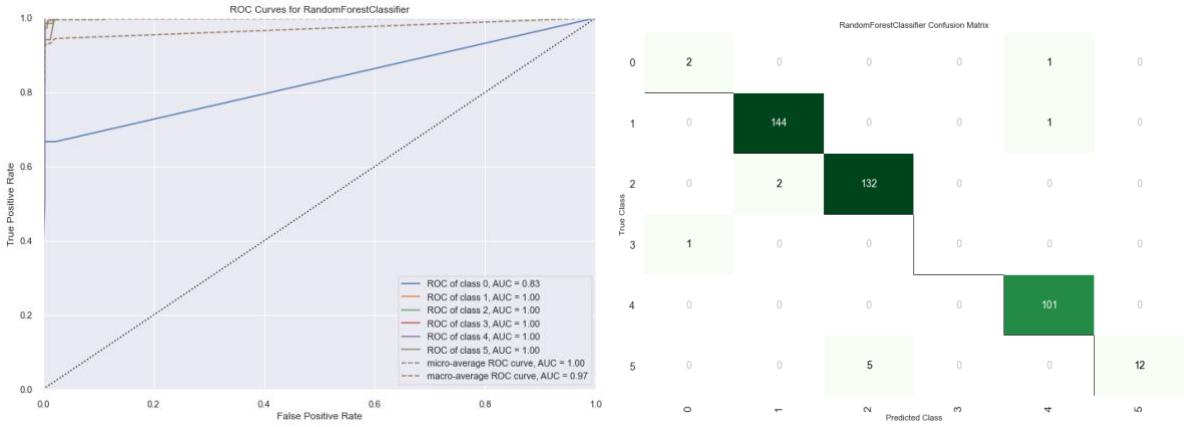


Figure 5-20. AUC and confusion matrix of Random Forests on unseen/test data of transformed location dataset.

The model analysis shows the fact that K-nearest neighbors and Random Forests still are not sub optimal and therefore misclassify positive and negative class instances in significant numbers. The Cat-boost and light gradient boosting are more reliable than logistic regression in terms of specificity and sensitivity. The tuned models seem to be slightly better than untuned counterparts, as they were optimized on AUC. The last section covers final predictions using ensemble of modes (Bagging, Boosting and Blender). The analysis covers specific versions (even though AUC and Confusion matrices have been plotted for all).

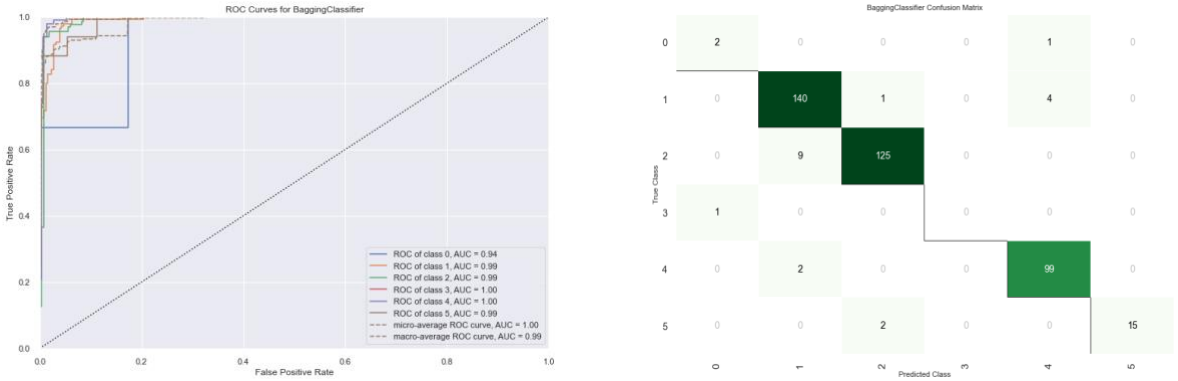


Figure 5-21. AUC and confusion matrix of Bagging classifier (ensemble of models) on unseen/test data of transformed location dataset.

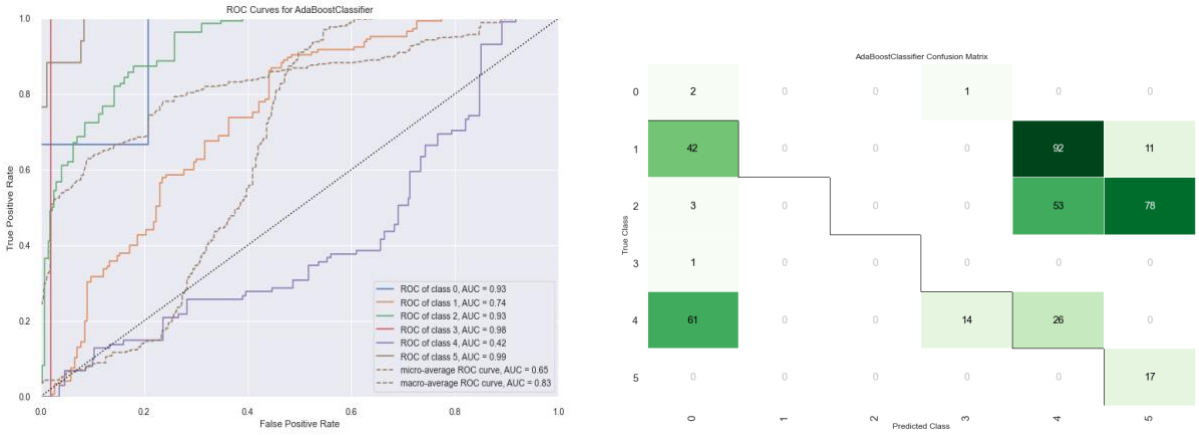


Figure 5-22. AUC and confusion matrix of Boosting Classifier (ensemble of models) on unseen/test data of transformed location dataset.

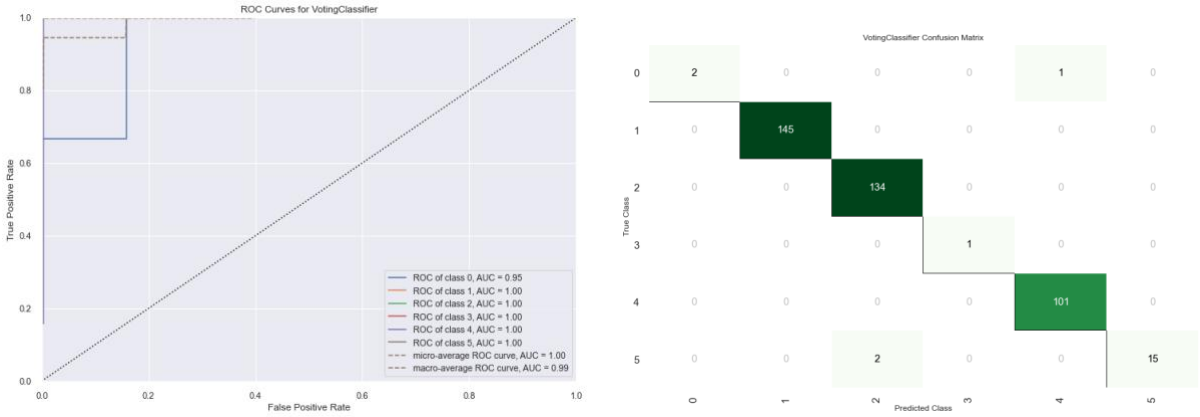


Figure 5-23.AUC and confusion matrix of Blender model on unseen/test data of transformed location dataset.

The bagged version of logistic regression and it's boosted version seem to perform worse than the tuned versions of logistic regression. The root cause of this could be attributed to unknown features occupying the feature space. Even the ensemble of models is not sufficient to cover the complexity of the environment if the classifier is unable to model the unknown variables in the domain. Due to this reason, a blender with combination of Logistic regression, Cat-boost classifier, K-nearest neighbors, Light gradient boosting classifier and Random forests is trained and tested to make final predictions on unknown data or test set which has not been used for modelling.

5.5 Conclusion

This concludes the experimental results and analysis section. The two phases regression and classification make this framework highly efficient combined with a long list of supervised learning algorithms. The evaluation of model has been carried out in depth to make better analysis of different phases of model generation. Often overlooked, dataset generation and preprocessing phase are very important steps in the framework which helps tune the model for different domains. The distance estimation and fingerprint intensity classification implemented in the proposed framework are a generalization of the improved indoor localization. This framework could serve to be a breakthrough in the upcoming era where SLAM (Simultaneous Localization and Mapping) has become important to map indoor areas eliminating the need to rely on GPS for accurate indoor positioning.

Chapter 6 Conclusions and Future Work

6.1 Conclusions

The research presented in thesis focused on improving state of art indoor localization via improved machine learning framework for indoor positioning and fingerprint intensity level classification. Indoor localization has become one of the crucial fields of research. The framework implemented means to serve as a base platform for further research. Dataset generation, preprocessing, adding spatial context to data, model generation and model interpretation are key elements of the framework. In addition to the framework, the ML-Ops is another key factor of this framework. Instead of focusing all the time on data preparation and preprocessing, focusing on improving machine learning models and gaining useful insights is highly encouraged. The optimization of models and their interpretation helps us gain useful insights of any indoor environment thereby reducing complexity of manual efforts.

6.2 Future Work

The framework is currently data-centric, and some improvements are needed to allow multiple machine learning models to perform well. The shift from data-centric to model-centric framework is proposed for future improvements, where significant noise can be handled by the framework and maintain accuracy at the same time. The use of neural models in the current framework could be biased since the dataset used is on a smaller scale. The generalization of framework to work with varying environment and interferences is the next best thing to be achieved. Convnets or Convolutional Neural Networks and modifying the framework to include image datasets is a suggested direction of research to reduce the localization error in indoor non-line-of-sight (NLoS) conditions. Last but not least, visualization of varying intensity levels in real time inference is a domain yet to be explored and researched upon. The appendix will cover the deployment phase which is still a work in progress.

Bibliography

Adewumi, O. G., Djouani, K. and Kurien, A. M. (2013) 'RSSI based indoor and outdoor distance estimation for localization in WSN', in *2013 IEEE International Conference on Industrial Technology (ICIT)*. *2013 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1534–1539. doi: 10.1109/ICIT.2013.6505900.

Afaneh, M. A. (no date) 'Bluetooth 5 & Bluetooth Low Energy: A Developer's Guide [2019]', *Novel Bits*. Available at: https://www.novelbits.io/bluetooth-5-developers-e-book/ (Accessed: 14 July 2021).

Arthi, K. and Lochana, A. S. R. (2019) 'Zone-based dual sub sink for network lifetime maximization in wireless sensor network', *Cluster Computing*, 22(6), pp. 15273–15283. doi: 10.1007/s10586-018-2563-7.

Atia, M. M., Noureldin, A. and Korenberg, M. J. (2013) 'Dynamic Online-Calibrated Radio Maps for Indoor Positioning in Wireless Local Area Networks', *IEEE Transactions on Mobile Computing*, 12(9), pp. 1774–1787. doi: 10.1109/TMC.2012.143.

Babcock University *et al.* (2017) 'Supervised Machine Learning Algorithms: Classification and Comparison', *International Journal of Computer Trends and Technology*, 48(3), pp. 128–138. doi: 10.14445/22312803/IJCTT-V48P126.

Bahl, P. and Padmanabhan, V. N. (2000) 'RADAR: an in-building RF-based user location and tracking system', in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, pp. 775–784 vol.2. doi: 10.1109/INFCOM.2000.832252.

Bloem, J. W. H. van and Schiphorst, R. (2011) 'Measuring the service level in the 2.4 GHz ISM band'. Available at: https://research.utwente.nl/en/publications/measuring-the-service-level-in-the-24-ghz-ism-band (Accessed: 14 July 2021).

Bose, A. and Foh, C. H. (2007) 'A practical path loss model for indoor WiFi positioning enhancement', in *2007 6th International Conference on Information, Communications Signal Processing*. *2007 6th International Conference on Information, Communications Signal Processing*, pp. 1–5. doi: 10.1109/ICICS.2007.4449717.

Brownlee, J. (2016a) 'Bagging and Random Forest Ensemble Algorithms for Machine Learning', *Machine Learning Mastery*, 21 April. Available at: https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/ (Accessed: 15 July 2021).

Brownlee, J. (2016b) ‘Boosting and AdaBoost for Machine Learning’, *Machine Learning Mastery*, 24 April. Available at: https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/ (Accessed: 15 July 2021).

Brownlee, J. (2020) ‘Stacking Ensemble Machine Learning With Python’, *Machine Learning Mastery*, 9 April. Available at: https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/ (Accessed: 15 July 2021).

Caire, G. and Shamai, S. (1999) ‘On the capacity of some channels with channel state information’, *IEEE Trans. Inf. Theory*. doi: 10.1109/18.782125.

Çolakoğlu, H. B. (2019) ‘A generalization of the Minkowski distance and a new definition of the ellipse’, *arXiv:1903.09657 [math]*. Available at: http://arxiv.org/abs/1903.09657 (Accessed: 15 July 2021).

Coronavirus Disease (COVID-19) Situation Reports (no date). Available at: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/situation-reports (Accessed: 14 July 2021).

Craw, S. (2017) ‘Manhattan Distance’, in Sammut, C. and Webb, G. I. (eds) *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer US, pp. 790–791. doi: 10.1007/978-1-4899-7687-1_511.

Crow, B. P. *et al.* (1997) ‘IEEE 802.11 Wireless Local Area Networks’, *IEEE Communications Magazine*, 35(9), pp. 116–126. doi: 10.1109/35.620533.

Dinh-Van, N. *et al.* (2017) ‘Indoor Intelligent Vehicle localization using WiFi received signal strength indicator’, in *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pp. 33–36. doi: 10.1109/ICMIM.2017.7918849.

Dokmanic, I. *et al.* (2015) ‘Euclidean Distance Matrices: Essential Theory, Algorithms and Applications’, *IEEE Signal Processing Magazine*, 32(6), pp. 12–30. doi: 10.1109/MSP.2015.2398954.

doodlemania2 (no date) *Azure IoT reference architecture - Azure Reference Architectures*. Available at: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot (Accessed: 16 July 2021).

Gohar, U. (2020) *How to use Residual Plots for regression model validation?, Medium*. Available at: https://towardsdatascience.com/how-to-use-residual-plots-for-regression-model-validation-c3c70e8ab378 (Accessed: 23 July 2021).

Gomez, C., Oller Bosch, J. and Paradells, J. (2012) ‘Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology’, *Sensors (Basel, Switzerland)*, 12, pp. 11734–53. doi: 10.3390/s120911734.

Grisetti, G. *et al.* (2010) ‘A tutorial on graph-based SLAM’, *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2, pp. 31–43. doi: 10.1109/MITS.2010.939925.

Grokking Machine Learning (no date) *Manning Publications*. Available at: https://www.manning.com/books/grokking-machine-learning (Accessed: 15 July 2021).

Guo, X. *et al.* (2018) ‘Indoor Localization by Fusing a Group of Fingerprints Based on Random Forests’, *IEEE Internet of Things Journal*, 5(6), pp. 4686–4698. doi: 10.1109/JIOT.2018.2810601.

Hartley, R. I. and Sturm, P. (1997) ‘Triangulation’, *Computer Vision and Image Understanding*, 68(2), pp. 146–157. doi: 10.1006/cviu.1997.0547.

ICML’04 tutorial on ROC analysis (no date). Available at: http://people.cs.bris.ac.uk/~flach/ICML04tutorial// (Accessed: 23 July 2021).

Imam, A., Kumar, V. and Srivastava, V. (2018) ‘Empirical predictions for the mechanical properties of Quaternary Cement Concrete’, *Journal of Structural Integrity and Maintenance*, 3(3), pp. 183–196. doi: 10.1080/24705314.2018.1492668.

Indoor location determination with RSSI (no date). Available at: https://kaggle.com/amirma/indoor-location-determination-with-rssi (Accessed: 16 July 2021).

Jiang, X. *et al.* (2018) ‘FSELM: fusion semi-supervised extreme learning machine for indoor localization with Wi-Fi and Bluetooth fingerprints’, *Soft Computing*, 22(11), pp. 3621–3635. doi: 10.1007/s00500-018-3171-4.

Kotsiantis, S. B., Zaharakis, I. D. and Pintelas, P. E. (2006) ‘Machine learning: a review of classification and combining techniques’, *Artificial Intelligence Review*, 26(3), pp. 159–190. doi: 10.1007/s10462-007-9052-3.

Lundberg, S. and Lee, S.-I. (2017) ‘A Unified Approach to Interpreting Model Predictions’, *arXiv:1705.07874 [cs, stat]*. Available at: http://arxiv.org/abs/1705.07874 (Accessed: 17 July 2021).

maggiesMSFT (no date) *Power BI documentation - Power BI*. Available at: https://docs.microsoft.com/en-us/power-bi/ (Accessed: 23 July 2021).

Malekpour, A., Ling, T. C. and Lim, W. C. (2008) ‘Location Determination Using Radio Frequency RSSI and Deterministic Algorithm’, in, pp. 488–495. doi: 10.1109/CNSR.2008.32.

National Marine Electronics Association - NMEA (no date). Available at: https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard (Accessed: 14 July 2021).

Norouzi, M., Fleet, D. and Salakhutdinov, R. (2012) ‘Hamming Distance Metric Learning’, in. *Advances in Neural Information Processing Systems*.

PyCaret — pycaret 2.2.0 documentation (no date). Available at: https://pycaret.readthedocs.io/en/latest/ (Accessed: 14 July 2021).

Rasmussen, C. and Ghahramani, Z. (2000) 'Occam's Razor', in *NIPS*. doi: 10.1201/9781420034639-138.

Robbins, H. (2007) 'A Stochastic Approximation Method'. doi: 10.1214/AOMS/1177729586.

Rocca, J. (2021) *Ensemble methods: bagging, boosting and stacking*, *Medium*. Available at: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205 (Accessed: 23 July 2021).

Sengar, P. P., Gaikwad, M. J. and Nagdive, A. S. (2020) 'Comparative Study of Machine Learning Algorithms for Breast Cancer Prediction', in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 796–801. doi: 10.1109/ICSSIT48917.2020.9214267.

Sturgess, B. N. and Carey, F. T. (1987) 'Trilateration', in Brinker, R. C. and Minnick, R. (eds) *The Surveying Handbook*. Boston, MA: Springer US, pp. 340–389. doi: 10.1007/978-1-4757-1188-2_11.

Subbu, K. P., Gozick, B. and Dantu, R. (2013) 'LocateMe: Magnetic-fields-based indoor localization using smartphones', *ACM Transactions on Intelligent Systems and Technology*, 4(4), p. 73:1-73:27. doi: 10.1145/2508037.2508054.

Subedi, S. and Pyun, J.-Y. (2017) 'Practical Fingerprinting Localization for Indoor Positioning System by Using Beacons', *Journal of Sensors*, 2017, p. e9742170. doi: 10.1155/2017/9742170.

Tang, J., Deng, C. and Huang, G.-B. (2016) 'Extreme Learning Machine for Multilayer Perceptron', *IEEE Transactions on Neural Networks and Learning Systems*, 27(4), pp. 809–821. doi: 10.1109/TNNLS.2015.2424995.

Townsend, K. (no date) *Introduction to Bluetooth Low Energy*, *Adafruit Learning System*. Available at: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction (Accessed: 14 July 2021).

Wang, Y. *et al.* (2018) 'WiFi Indoor Localization with CSI Fingerprinting-Based Random Forest', *Sensors*, 18(9), p. 2869. doi: 10.3390/s18092869.

Witham, O. *et al.* (2019) 'Batteryless Wireless Water Leak Detection System', in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*. *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1–4. doi: 10.1109/SmartNets48225.2019.9069789.

Wu, K. *et al.* (2013) 'CSI-Based Indoor Localization', *IEEE Transactions on Parallel and Distributed Systems*, 24(7), pp. 1300–1309. doi: 10.1109/TPDS.2012.214.

Xiaodong, G. *et al.* (2018) 'Indoor localization method of intelligent mobile terminal based on BIM', in *2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS). 2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS)*, pp. 1–9. doi: 10.1109/UPINLBS.2018.8559731.

Ying, X. (2019) 'An Overview of Overfitting and its Solutions', *Journal of Physics: Conference Series*, 1168, p. 022022. doi: 10.1088/1742-6596/1168/2/022022.

Zhao, Z. *et al.* (2017) 'NaviLight: Indoor localization and navigation under arbitrary lights', in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9. doi: 10.1109/INFOCOM.2017.8057184.

Zhou, R. *et al.* (2017) 'Device-Free Presence Detection and Localization With SVM and CSI Fingerprinting', *IEEE Sensors Journal*, 17(23), pp. 7990–7999. doi: 10.1109/JSEN.2017.2762428.

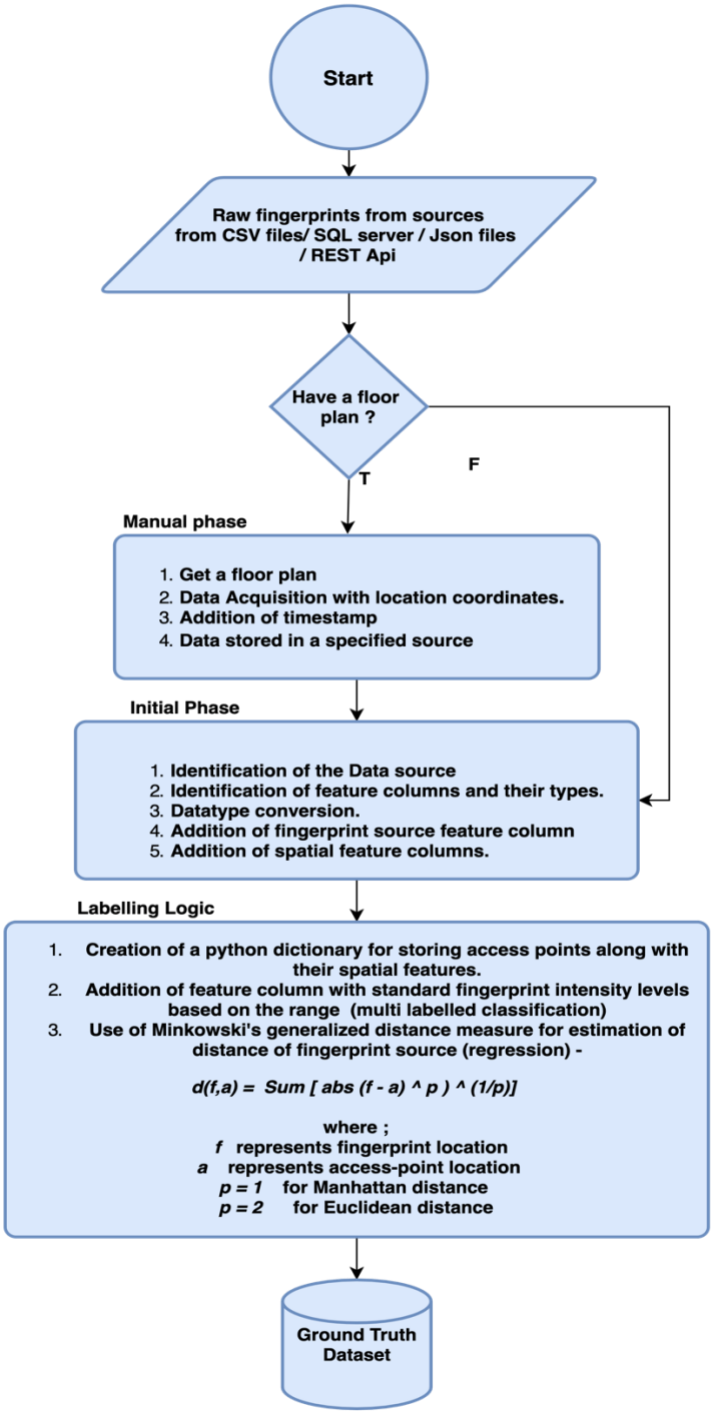
Zhu, X. *et al.* (2020) 'Indoor Intelligent Fingerprint-Based Localization: Principles, Approaches and Challenges', *IEEE Communications Surveys Tutorials*, 22(4), pp. 2634–2657. doi: 10.1109/COMST.2020.3014304.

Appendix

A. Dataset Generation code

This section describes the dataset generation code. The major implementation has been done using python 3.6, anaconda and use of some major libraries including NumPy, Matplotlib, pandas, Pycaret, Scikit, Pyodbc, Cat-boost and Shap. NumPy is used for manipulation of multidimensional array objects, sorting and shape manipulations. Matplotlib is used for plotting various useful visualizations. Pandas provide an efficient way to use data structures and data analysis tools for python. Pycaret is the low-code framework which helps someone with machine learning pipeline setup without much hassle and prior knowledge about machine learning pipelines. Scikit-learn covers major regression and classification implementations. Cat-boost is an open-source library for applying gradient boosting on decision trees. Shap is used for interpretation of models after they are trained. Dataset generation code makes use of Minkowski's distance metric for Euclidean and Manhattan distance metrics either of which can be used for distance estimation using fingerprints. The code also integrates spatial context to existing raw fingerprint data in form of grids or latitude-longitudes.

Flowchart


Start
Raw fingerprints from sources from CSV files/ SQL server / Json files / REST Api
Have a floor plan ?
T
F
Manual phase
1. Get a floor plan
2. Data Acquisition with location coordinates.
3. Addition of timestamp
4. Data stored in a specified source
Initial Phase
1. Identification of the Data source
2. Identification of feature columns and their types.
3. Datatype conversion.
4. Addition of fingerprint source feature column
5. Addition of spatial feature columns.
Labelling Logic
1. Creation of a python dictionary for storing access points along with their spatial features.
2. Addition of feature column with standard fingerprint intensity levels based on the range (multi labelled classification)
3. Use of Minkowski's generalized distance measure for estimation of distance of fingerprint source (regression) -
d(f,a) = Sum [abs (f - a) ^ p) ^ (1/p)]
where ;
f represents fingerprint location
a represents access-point location
p = 1 for Manhattan distance
p = 2 for Euclidean distance
Ground Truth Dataset

Code

```
########################## Installation of required packages ####################
!pip install numpy
!pip install matplotlib
!pip install pandas
!pip install pycaret
!pip install scikit-optimize
!pip install catboost
!pip install pyodbc
  !pip install shap
############################## Import packages ################################
  from numpy import vstack
  from numpy import sqrt
  from pandas import read_csv
  import pyodbc
  import getpass
  import pandas as pd
  import numpy as np
  from datetime import datetime
  from dateutil import tz
  from pycaret.utils import enable_colab
  import pandas as pd
  enable_colab()
  %matplotlib inline
  import matplotlib.pyplot as plt
  import numpy as np
  import seaborn as sns
  import pandas as pd
  import scipy as sc
  sns.set()
  import matplotlib
  from pycaret.regression import *
  matplotlib.rcParams['figure.figsize'] = [12, 8]
```

```
from pycaret.datasets import get_data
########################## Dataset Class ####################################
class Dataset():
########################## Source of data definition ##########################
def display(self):
print("Choose source of your indoor location data:")
print("1-> SQL server")
print("2-> CSV file")
print("3-> Json file")
print("4-> Via API call")

########################## SQL server connection using pyodbc package #########
def sqlapi(self,server,database,username,password,table_name):
conn = pyodbc.connect('DRIVER={SQL
Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)
cursor = conn.cursor()
query = "SELECT * from "+ table_name + " ;"
df = pd.read_sql(query,conn)
return df
########################## CSV (Comma-Separated Values) files ################
def csv_files(self,filename,s):
df = pd.read_csv(filename,sep=s)
return df
############################### JSON files ###################################
def read_json(self,file):
df = pd.read_json(file)
df.info()
return df
############################### API Call  ####################################
def api_call(self,URL):
df = pd.read_json(URL)
return df
##################### Datatype conversion for numeric/datetime types##############
def datatype_convert(self,df,col):
choice = int(input("Press 1 for numeric / Press 2 for Datetime"))
if(choice == 1):
```

```
df[col] = pd.to_numeric(df[col])
elif(choice == 2):
try:
df[col] = pd.to_datetime(df[col], format="%Y-%d-%m %H:%M:%S")
except:
df[col] = pd.to_datetime(df[col])
return df

def data_sources(self,src):
if(src == 1):
server = input("Enter the server name: ")
database = input("Enter the database name: ")
username = input("Enter the username: ")
password = getpass.getpass("Enter the password: ")
table_name = input("Enter the table name you want to query out ")
df = self.sqlapi(server,database,username,password,table_name)
elif(src == 2):
file = input("Enter the file path ")
s = input("Enter the separator if any ")
df = self.csv_files(file,s)
elif(src == 3):
file = input("Enter the json file path ")
df = self.read_json(file)
elif(src == 4):
url = input("Enter the REST URL ")
df = self.api_call(url)
else:
print("Sorry Invalid choice !! ")
return df

############################ Saving files ######################################
def save_file(self,df):
print("Location Dataset: \n",df)
file_path_tf = input('Enter the file you want to store the transformed dataset: ')
file_name = input("Enter the file name with .csv extension")
```

```
    df.to_csv(file_path_tf+'/'+file_name,index=False)

    ######################### Bluetooth Address / Sensor address ###################
    def check_sensor_bluetooth(self,df):
    hv_sensor_col = input("Do you have a sensor/bluetooth  column in your dataset? ")
    if(hv_sensor_col == "no"):
    data_col = input("Enter the name of sensor data column: ")
    address = input("Enter the bluetooth address:")
    df['Sensor'] = np.where(df[data_col] == address,"Sensor Device","Bluetooth Device")
    return df

    ################# Initial pass of processing raw data without spatial context###########
    def initial_pass(self):
    self.display()
    src = int(input("Enter the source of the data: (make sure path doesn't contain whitespaces or special
symbols)"))
    df = self.data_sources(src)
    print("\n Data loaded on a dataframe.....")
    col_names = []
    count = len(df.columns)
    print("\n Columns present in current dataset.... :",count)
    col_names.append(df.columns)
    #df = pd.DataFrame(df,columns = col_names)
    df.convert_dtypes().dtypes
    print("\n Preview of Data: \n",df.head(50))
    print("\n Summary of data: \n",df.info())
    d_type_conv = input("Do you want to convert Datatype of any specific column (yes/no) :")
    while(d_type_conv == "yes"):
    col = input("Enter the name of column whose datatype you want to change: ")
    self.datatype_convert(df,col)
    d_type_conv = input("Do you want to change datatype of one more column(yes/no): ")
    print("\n Summary of data after conversion \n",df.info())
    self.check_sensor_bluetooth(df)
    rssi_col = input("Enter the fingerprint column in the dataset :")

    ############### RSSI fingerprint intensity binning based on standard scales ##########
```

```
def rssi_levels(df):
if df[rssi_col] < -100:
return "No Signal"
elif df[rssi_col] >= -100 and df[rssi_col] <= -90:
return "Very low"
elif df[rssi_col] >= -90 and df[rssi_col] <= -80:
return "Low"
elif df[rssi_col] >= -80 and df[rssi_col] <= -70:
return "Good"
elif df[rssi_col] >= -70 and df[rssi_col] <= -60:
return "Very good"
elif df[rssi_col] >= -60 and df[rssi_col] <= -50:
return "Excellent"
elif df[rssi_col] > -50:
return "Outstanding"
df["RSSI_Intensity"] = df.apply(rssi_levels,axis=1)
print("\n Preview of data before proceeding forward .....")
return df

ds = Dataset()
df = ds.initial_pass()
################################### Location Class ##############################
class Location():
def __init__(self,df):
self.df = df

def location(self,df):
loc_data = input("Do you have manual location coordinates in your dataset (yes/no) ")
if(loc_data == "no"):
how_many = int(input("Enter the count of locations you want to enter: "))
id = input("Enter the beacon identifier column name: ")
x_loc = input("Enter the x location col_name: ")
y_loc = input("Enter the y location col_name: ")
z_loc = input("Enter the floor location col_name: ")
loc_name_col = input("Enter the location name col_name:")
```

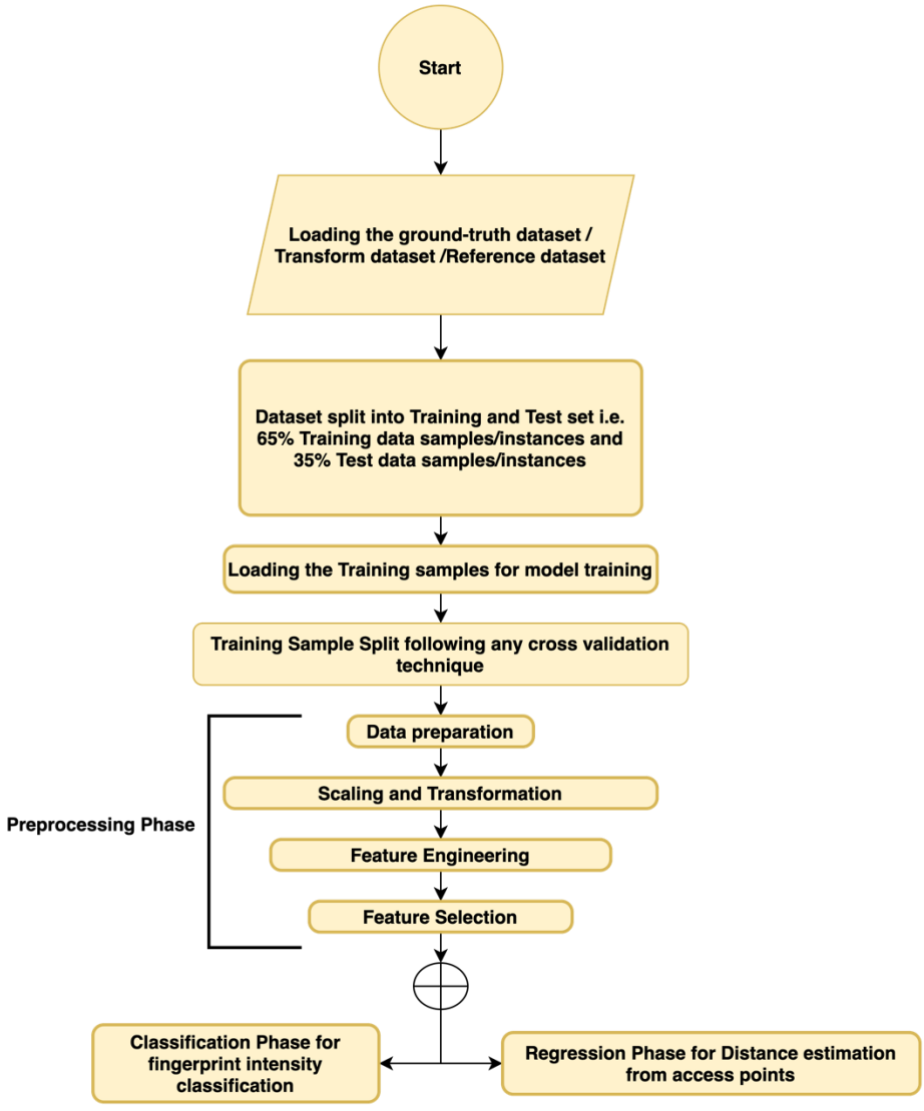
```
    for i in range(how_many):
    idn = input("Enter the beacon id: ")
    loc_name = input("Enter the location name: ")
    x = float(input("Enter the x / latitude coordinate / grid coordinate"))
    y = float(input("Enter the y / longitude coordinate / grid coordinate"))
    z = float(input("Enter the z / floor coordinate / grid coordinate"))
    df.loc[df[id] == idn,x_loc] = x
    df.loc[df[id] == idn,y_loc] = y
    df.loc[df[id] == idn,z_loc] = z
    df.loc[df[id] == idn,loc_name_col] = loc_name
    return df

    loc_ds = loc1.location(df)
    ds.save_file(loc_ds) ### Save file with some name
    dataset_file = ('location_dataset_thesis_final.csv') ### retrieve the file with the same name
    data = pd.read_csv(dataset_file)
    print(data.count)
    dataset = data.copy()

    #########Generalizing all distance measures in form of minowski's distance metric#####
    p = int(input("Enter 1 for Manhattan distance measure and 2 for Euclidean distance measure"))

    ######### Location of the Access point or Bluetooth gateway ######################

    ap_coordinates = {"1":{43.476462,-80.570871,2},"2":{43.476473,-80.570879,2},"3":{43.4764129, -
80.5707663,2},"4":{43.4764791, -80.5710245,2}}
    x_loc = input("Enter the x location col_name (sensors)")
    y_loc = input("Enter the y location col_name (sensors)")
    z_loc = input("Enter the floor location col_name (sensors)")
    access_point_col = input("Enter the name of access point column(make sure it's a string column")
    g = dataset.groupby([x_loc,y_loc,z_loc,access_point_col])
    def dist(df):
    ap_coords = ap_coordinates[df[access_point_col].iloc[0]]
    x, y, z = ap_coords
    df["distance"] = pow((np.abs((df[x_loc] - x)) ** p + np.abs((df[y_loc] - y)) ** p + np.abs((df[z_loc] - z)) **
p),(1/p))
```

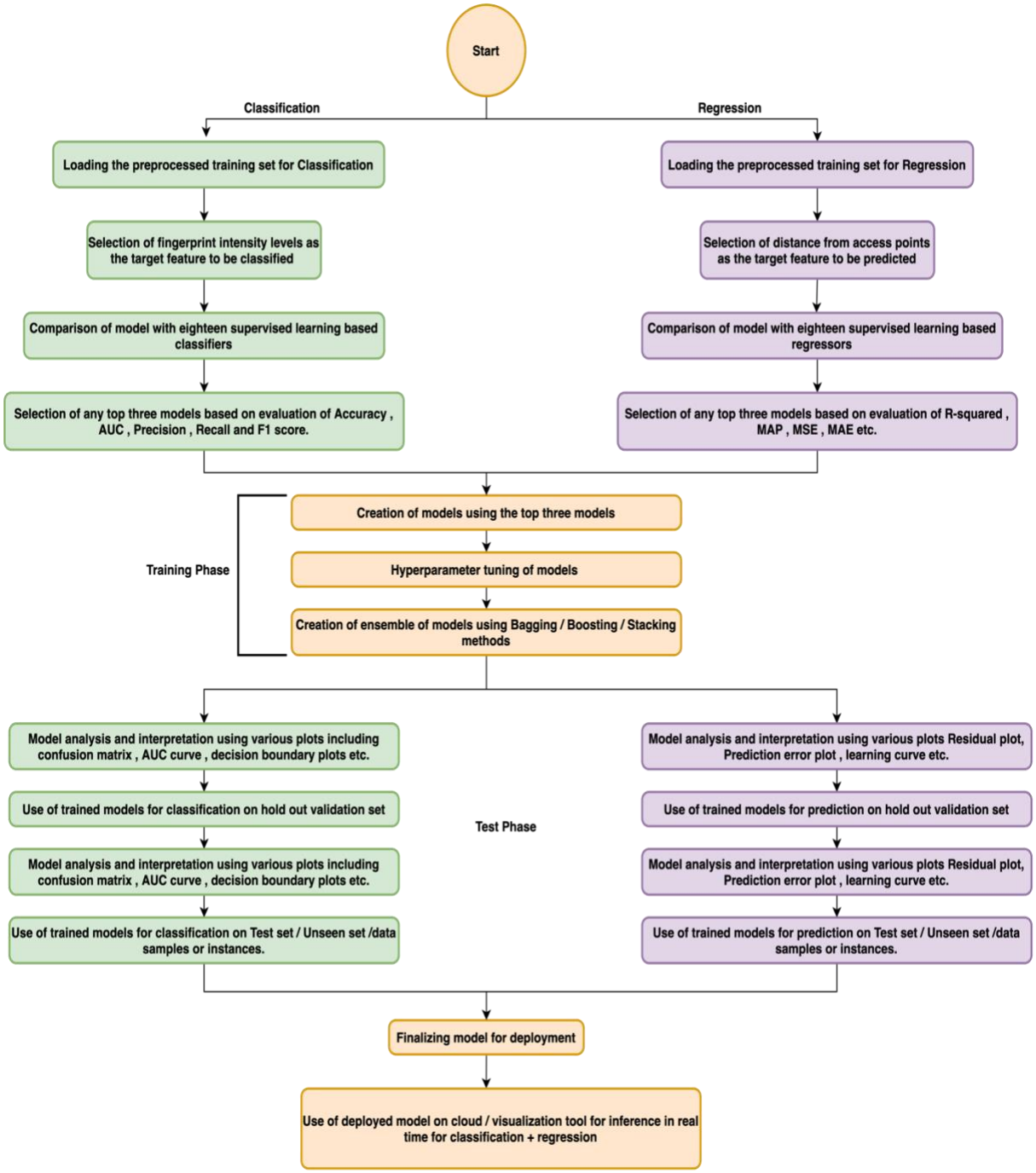
```
return df
dataset  = g.apply(dist)
```

B. Model generation code (Classification plus Regression)

This section includes the code for using the transformed dataset for classification and regression. The pipeline includes loading of dataset, comparison of models, model creation, model training and predictions on holdout/validation set and final predictions on unseen data or test set.

Flowchart




Start
Classification
Regression
Loading the preprocessed training set for Classification
Loading the preprocessed training set for Regression
Selection of fingerprint intensity levels as the target feature to be classified
Selection of distance from access points as the target feature to be predicted
Comparison of model with eighteen supervised learning based classifiers
Comparison of model with eighteen supervised learning based regressors
Selection of any top three models based on evaluation of Accuracy , AUC , Precision , Recall and F1 score.
Selection of any top three models based on evaluation of R-squared , MAP , MSE , MAE etc.
Training Phase
Creation of models using the top three models
Hyperparameter tuning of models
Creation of ensemble of models using Bagging / Boosting / Stacking methods
Model analysis and interpretation using various plots including confusion matrix , AUC curve , decision boundary plots etc.
Model analysis and interpretation using various plots Residual plot, Prediction error plot , learning curve etc.
Use of trained models for classification on hold out validation set
Use of trained models for prediction on hold out validation set
Test Phase
Model analysis and interpretation using various plots including confusion matrix , AUC curve , decision boundary plots etc.
Model analysis and interpretation using various plots Residual plot, Prediction error plot , learning curve etc.
Use of trained models for classification on Test set / Unseen set /data samples or instances.
Use of trained models for prediction on Test set / Unseen set /data samples or instances.
Finalizing model for deployment
Use of deployed model on cloud / visualization tool for inference in real time for classification + regression

Code

```
######################## import the transformed dataset file ####################
from imblearn.over_sampling import *
dataset_file = ('.//transformed_thesis_final.csv')
data_read = pd.read_csv(dataset_file)
print(data_read.count)
dataset = data_read.copy()

######################### Check the dataset shape ################################
print("Shape of the dataset (observations, features): ",dataset.shape)

######################### Train – Test dataset split ##############################
data = dataset.sample(frac=0.65,random_state=600)
data_unseen = dataset.drop(data.index)
data.reset_index(inplace=True,drop=True)
data_unseen.reset_index(inplace=True,drop=True)
print('Data for Modeling:' + str(data.shape))
print('Unseen Data for Predictions:' + str(data_unseen.shape))

######################## Checking the fingerprint intensity counts #################
data.RSSI_Intensity.value_counts()
data["RSSI_Intensity"].value_counts().plot.bar(legend=None)

############################ Preprocessing Stage ################################
from pycaret.classification import *
adasyn1 = ADASYN(sampling_strategy='minority')
exp2 = setup(data = data, target =
'RSSI_Intensity',session_id=111,preprocess=True,imputation_type='iterative',iterative_imputation_iters=1
0,date_features=['timestamp'],normalize=True,normalize_method='zscore',transformation=True,transform
ation_method='quantile',log_plots=True,log_experiment = True,use_gpu=True,experiment_name =
'Fingeprint Multilabel classification')

######################### Comparison of models ################################
best_model  = compare_models()
```

```
    ########################## Classifier Model creation ############################
    lightbgm = create_model('lightgbm')
    cat = create_model('catboost')
    lr = create_model('lr')
    rf = create_model('rf')
    knn = create_model('knn')
    tuned_lightbgm = tune_model(lightbgm, optimize = 'AUC') #default is 'Accuracy'
    tuned_cat = tune_model(cat, optimize = 'AUC') #default is 'Accuracy'
    tuned_lr = tune_model(lr, optimize = 'AUC') #default is 'Accuracy'
    tuned_knn = tune_model(knn, optimize = 'AUC') #default is 'Accuracy'
    tuned_rff = tune_model(rf, optimize = 'AUC') #default is 'Accuracy'
    bagged_lightbgm = ensemble_model(tuned_lightbgm, method = 'Bagging')
    bagged_lr = ensemble_model(tuned_lr, method = 'Bagging')
    bagged_rf = ensemble_model(tuned_rff, method = 'Bagging')
    bagged_knn = ensemble_model(tuned_knn, method = 'Bagging')
    bagged_cat = ensemble_model(tuned_cat,method='Bagging')
    boosted_lightbgm = ensemble_model(tuned_lightbgm, method = 'Boosting')
    boosted_lr = ensemble_model(tuned_lr, method = 'Boosting')
    boosted_rf = ensemble_model(tuned_rff, method = 'Boosting')
    blender_normal = blend_models(estimator_list = [knn,rf,lr,cat,lightbgm], method = 'soft')
    stacker = stack_models(estimator_list = [lr,knn,rf,cat,lightbgm])

########################## Predictions on validation/hold-out set ##################
    predicted_lr = predict_model(lr)
    predicted_lightbgm = predict_model(lightbgm)
    predicted_cat = predict_model(cat)
    predicted_knn = predict_model(knn)
    predicted_rf = predict_model(rf)
    predicted_tuned_lr = predict_model(tuned_lr)
    predicted_tuned_lightbgm = predict_model(tuned_lightbgm)
    predicted_tuned_cat = predict_model(tuned_cat)
    predicted_tuned_knn = predict_model(tuned_knn)
    predicted_tuned_rf = predict_model(tuned_rff)
    predicted_bagged_lr = predict_model(bagged_lr)
    predicted_bagged_lightbgm = predict_model(bagged_lightbgm)
```

```
predicted_bagged_cat = predict_model(bagged_cat)
predicted_bagged_knn = predict_model(bagged_knn)
predicted_bagged_rf = predict_model(bagged_rf)
predicted_boosted_lr = predict_model(bagged_lr)
predicted_boosted_lightbgm = predict_model(boosted_lightbgm)
predicted_boosted_cat = predict_model(boosted_cat)
predicted_boosted_rf = predict_model(boosted_rf)
predicted_blender = predict_model(blender_normal)
predicted_stacked = predict_model(stacker)

########################## Predictions on unseen test sets ######################
## run this only once
predicted_lr = predict_model(lr,data=data_unseen)
predicted_lightbgm = predict_model(lightbgm,data=data_unseen)
predicted_cat = predict_model(cat,data=data_unseen)
predicted_knn = predict_model(knn,data=data_unseen)
predicted_rf = predict_model(rf,data=data_unseen)
predicted_tuned_lr = predict_model(tuned_lr,data=data_unseen)
predicted_tuned_lightbgm = predict_model(tuned_lightbgm,data=data_unseen)
predicted_tuned_cat = predict_model(tuned_cat,data=data_unseen)
predicted_tuned_knn = predict_model(tuned_knn,data=data_unseen)
predicted_tuned_rf = predict_model(tuned_rff,data=data_unseen)
predicted_bagged_lr = predict_model(bagged_lr,data=data_unseen)
predicted_bagged_lightbgm = predict_model(bagged_lightbgm,data=data_unseen)
predicted_bagged_cat = predict_model(bagged_cat,data=data_unseen)
predicted_bagged_knn = predict_model(bagged_knn,data=data_unseen)
predicted_bagged_rf = predict_model(bagged_rf,data=data_unseen)
predicted_boosted_lr = predict_model(bagged_lr,data=data_unseen)
predicted_boosted_lightbgm = predict_model(boosted_lightbgm,data=data_unseen)
predicted_boosted_cat = predict_model(boosted_cat,data=data_unseen)
predicted_boosted_rf = predict_model(boosted_rf,data=data_unseen)
predicted_blender = predict_model(blender_normal,data=data_unseen)
predicted_stacked = predict_model(stacker,data=data_unseen)

########################## Save the predictions ##################################
```

```
    predicted_lr.to_csv('.//predictions_lr.csv')
    predicted_lightbgm.to_csv('.//predictions_lightbgm.csv')
    predicted_cat.to_csv('.//predictions_cat.csv')
    predicted_knn.to_csv('.//predictions_knn.csv')
    predicted_rf.to_csv('.//predictions_rf.csv')
    predicted_tuned_lr.to_csv('.//predictions_tuned_lr.csv')
    predicted_tuned_lightbgm.to_csv('.//predictions_tuned_lightbgm.csv')
    predicted_tuned_cat.to_csv('.//predictions_tuned_cat.csv')
    predicted_tuned_knn.to_csv('.//predictions_tuned_knn.csv')
    predicted_tuned_rf.to_csv('.//predictions_tuned_rf.csv')
    predicted_bagged_lr.to_csv('.//predictions_bagged_lr.csv')
    predicted_bagged_lightbgm.to_csv('.//predictions_bagged_lightbgm.csv')
    predicted_bagged_cat.to_csv('.//predictions_bagged_cat.csv')
    predicted_bagged_knn.to_csv('.//predictions_bagged_knn.csv')
    predicted_bagged_rf.to_csv('.//predictions_bagged_rf.csv')
    predicted_boosted_lr.to_csv('.//predictions_boosted_lr.csv')
    predicted_boosted_lightbgm.to_csv('.//predictions_boosted_lightbgm.csv')
    predicted_boosted_cat.to_csv('.//predictions_boosted_cat.csv')
    predicted_boosted_rf.to_csv('.//predictions_boosted_rf.csv')
    predicted_blender.to_csv('.//predictions_blender.csv')
    predicted_stacked.to_csv('.//predictions_stacked.csv')

############################### Saving/Loading trained models ####################
    save_model(lr,'logistic_regression')
    save_model(lightbgm, 'lightbgm')
    save_model(cat, 'catboost')
    save_model(knn,'knn')
    save_model(rf,'rf')
    save_model(tuned_lr,'tuned_lr')
    save_model(tuned_cat,'tuned_cat')
    save_model(tuned_knn,'tuned_knn')
    save_model(tuned_rff,'tuned_rf')
    save_model(tuned_lightbgm,'tuned_lightbgm')
    save_model(bagged_lr,'bagged_lr')
    save_model(bagged_cat,'bagged_cat')
```

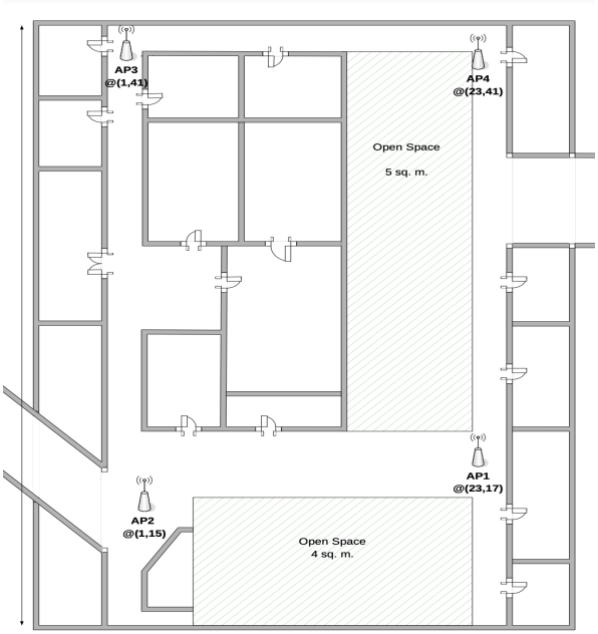
```
save_model(bagged_rf,'bagged_rf')
save_model(bagged_knn,'bagged_knn')
save_model(bagged_lightbgm,'bagged_lightbgm')
save_model(boosted_lr,'boosted_lr')
save_model(boosted_cat,'boosted_cat')
save_model(boosted_rf,'boosted_rf')
save_model(boosted_lightbgm,'boosted_lightbgm')
save_model(blender_normal,'blender_normal')
save_model(stacker,'stacking_model')
#####use this only when loading models
# lr = load_model('logistic_regression')
# lightbgm = load_model('lightbgm')
# cat = load_model('catboost')
# knn = load_model('knn')
# rf = load_model('rf')
# tuned_lr = load_model('tuned_lr')
# tuned_cat = load_model('tuned_cat')
# tuned_knn = load_model('tuned_knn')
# tuned_rff = load_model('tuned_rf')
# tuned_lightbgm = load_model('tuned_lightbgm')
# bagged_lr = load_model('bagged_lr')
# bagged_cat = load_model('bagged_cat')
# bagged_rf = load_model('bagged_rf')
# bagged_knn = load_model('bagged_knn')
# bagged_lightbgm = load_model('bagged_lightbgm')
# boosted_lr = load_model('boosted_lr')
# boosted_cat = load_model('boosted_cat')
# boosted_rf = load_model('boosted_rf')
# boosted_lightbgm = load_model('boosted_lightbgm')
# blender_normal = load_model('blender_normal')
# stacker = load_model('stacking_model')

######### Regression Phase for distance / coordinates estimation ##########################
from pycaret.regression import *
from pycaret.datasets import get_data
```

```
from imblearn.over_sampling import *
dataset_file = ('.//transformed_thesis_final.csv')
data = pd.read_csv(dataset_file)
print(data.count)
dataset = data.copy()

######################## Training/Test set split ######################################
data = dataset.sample(frac=0.65,random_state=300)
data_unseen = dataset.drop(data.index)
data.reset_index(inplace=True,drop=True)
data_unseen.reset_index(inplace=True,drop=True)
print('Data for Modeling:' + str(data.shape))
print('Unseen Data for Predictions:' + str(data_unseen.shape))

########################## Preprocessing Phase ####################################
from pycaret.regression import *
adasyn1 = ADASYN(sampling_strategy='minority')
regression_set = setup(data = data, target = 'distance',
session_id=123,preprocess=True,imputation_type='iterative',
iterative_imputation_iters=20,numeric_imputation='mean',
normalize = True,log_plots=True,
remove_multicollinearity = True, multicollinearity_threshold = 0.85,
use_gpu=True,experiment_name = 'Distance_estimation')

###################### Comparison of models ######################################
best_reg_model = compare_models()

###################### Model Creation ############################################
dt = create_model('dt',fold=5)
cat = create_model('catboost',fold=5)
knn = create_model('knn',fold=5)
tuned_knn = tune_model(knn)
bagged_knn = ensemble_model(knn)
boosted_knn = ensemble_model(knn,method='Boosting')
blender = blend_models(estimator_list=[cat,knn,dt])
stacker = stack_models(estimator_list=[cat,knn,dt])
```

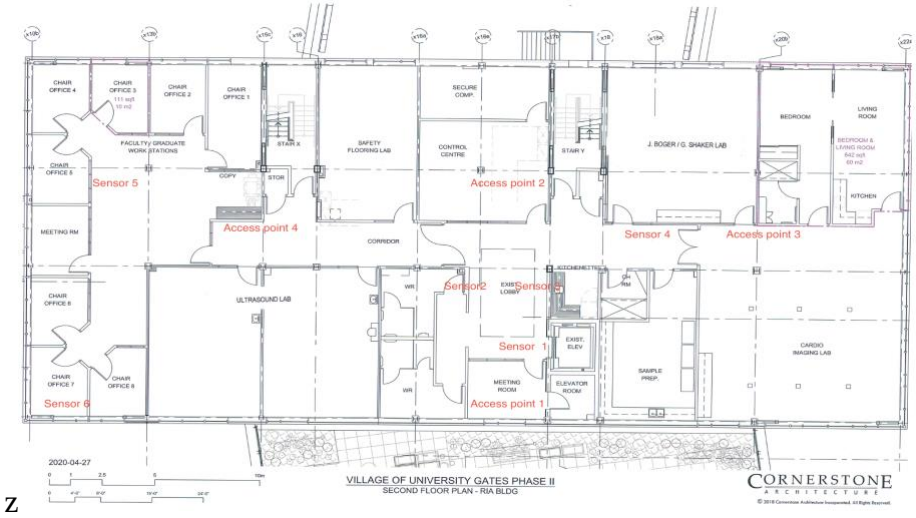
```
    ###################### Model interpretation and Creation ###############################
    interpret_model(dt)
    plot_model(dt)

    ###################### Predictions of model on holdout or validation set ###################
    predict_model(cat)
    predict_model(knn)
    predict_model(dt)
    predict_model(blender)
    predict_model(stacker)
################################ Saving of Models ####################################
    save_model(cat,'catboost regressor')
    save_model(dt,'Decision tree regressor')
    save_model(knn,'KNN regressor')
    save_model(blender,'Voting regressor')
    save_model(stacker,'Stacking regressor')

    ############################ Test set distance estimation ##############################
    data_unseen['access_point'] = data_unseen['access_point'].astype(str)
    fig = plt.figure(figsize=(8, 20))
    from itertools import product
    axs = fig.subplots(4,2)
    for pair, ax in zip(product((1,2), ("1","2","3","4")), axs.flatten()):
    (floor, access_point) = pair
    mask = (data_unseen.floor == floor) & (data_unseen.access_point == access_point)
    signal = data_unseen[mask][["RSSI", "distance"]]
    ax.plot(signal.distance, signal.RSSI, '.')
    ax.set_ylabel("RSSI")
    ax.set_title("Floor: %s AP: %s" %(floor, access_point))

    ############################ UI server running to collect all data and plots ###############
    !mlflow ui
```

C. Floor plans used for Dataset generation

Two different floor plans were utilized for creation of regression and classification framework and evaluation. The first floor plan has been used for reference dataset by author(Malekpour, Ling and Lim, 2008). The second-floor plan has been used for the transformation of raw location coordinates to a location dataset with fingerprint intensity levels. Both the floor plans are attached below –



Floor Plan used by author for reference dataset generation



Z

Floor plan used for transformed dataset (RIA building, University of Waterloo, Ontario,Canada)

D. Deployment Phase

This section covers a fraction of deployment phase which is still a work in progress. The deployment of trained models is done using Microsoft's Power BI which is a business analytics platform. The analytics solution is used for visualizing data and driving insight to actions. The further scope of the project is to embed the report on an application or a website. The proposed framework in the deployment phase is the use of an automated machine learning tool (Auto-ML) for indoor localization. This allows data analysts and data scientists to build machine learning models with efficiency and ensures efficiency of a model. The outcome of the Auto-ML framework is to use the best model based on performance criteria. To reduce the work of training and building using Auto-ML tool, the model generation code is used for training and saving the trained models. These trained models are then deployed on Power BI to ensure predictions/inference/classification in real time (maggiesMSFT, 2021). Following are some of the steps used for deployment of model on Power BI –

1. Setting up the environment using an Anaconda tool -

   conda create –name thesis python=3.6 (3.7 is the preferred one for deployment)

2. Activation of the virtual environment created in the previous step using –

   Conda activate thesis

3. Installation of the Pycaret library using pip manager

   Pip install Pycaret[full]

4. Loading the transformed dataset via any data source (SQL server / CSV files / JSON / API source)

RSSI	MAC	Data	message_id	timestamp	client	access_point	Sensor	RSSI_Intensity	lat	long
-70	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	8415baa5fa7a4d84b933b0640ecdca23	6/22/2021 5:44:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-70	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	4f69df2dc0fb40ec8dd8654d0c0c6ea4	6/22/2021 5:44:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-72	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	3d784722ecbd4c0384752aefb244d62b	6/22/2021 5:27:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-71	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	0f04f1cafa734d4c89f22c0e9b0cd8dd	6/22/2021 5:45:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-71	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	93b33b1fbce24eb98ace8ffe8788608b	6/22/2021 5:46:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-71	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	309b0848233a40aaa3d9f08ff4584a87	6/22/2021 5:47:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-70	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	542ac5f60e0f408a938849e82331fd0b	6/22/2021 5:40:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-70	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	d7b91cf132df4de380ef65b7c0985743	6/22/2021 5:40:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-70	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	11b36e46261248c3ba29d5005b6fd718	6/22/2021 5:41:00 PM	other	3	Sensor D			
-71	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	b75b0c5f749242d9b4faacc4e479f2c7	6/22/2021 5:42:00 PM	other	3	Sensor D			
-71	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	4d6ed2afbd38479a9912de2d9571882c	6/22/2021 5:42:00 PM	other	3	Sensor D			
-70	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	239fe272b3bf4cbc84b6e292d4679700	6/22/2021 5:44:00 PM	other	3	Sensor D			
-71	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	39826150141b45b19935880a2f5fd159	6/22/2021 5:45:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-71	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	44210a6d703e4d9a9cc80115d8f2f587	6/22/2021 5:48:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-76	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	58f841b301254b53b74766dafed2a14f	7/6/2021 2:54:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-73	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	2f061a40a09c493394bb93046e4ec4d5	7/6/2021 2:54:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-77	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	356e5962a752409aaebc91be52166675	7/6/2021 2:54:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-75	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	0d0972cd9dd547c791367e553454690c	7/6/2021 2:55:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-77	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	6a0f2d2248db40b5bea7869effa4d904	7/6/2021 2:55:00 PM	other	3	Sensor Device	Good	43.476447	-80.57
-73	F2:28:C0:0C:04:CA	0201041AFFFFFF0215AACC00AA55EE7755AACC00AA55E	3b0b0c58aec24726ae447d0d5cc4d617	7/6/2021 2:55:00 PM	other	3	Sensor Device	Good	43.476447	-80.57

F2:28:C0:0C:04:CA

Name 'transformed_thesis_final
Storage mode Import
Data refreshed 2021-07-19, 10:07:46 a.m.

5. Running python script

   Code

```
############## load functions from regression module ################################
from pycaret.classification import load_model, predict_model

############# load the saved model in a variable#####################################
model = load_model('C:/Users/aquasensing/Desktop/data repo/blender_normal')

############# Classification of fingerprint intensity levels ##############################
dataset = predict_model(model, data=dataset)
```

6. A column label will appear in the dataset with the classifications.

RSSI_Intensity
Good
Good

Label
Good
Good

7. The labels generated in form of classification are used for RSSI Intensity plots using custom visualizations in Power BI –

This concludes the appendix section. The proposed framework with dataset generation, model generation and real time inference-visualization is the apt way for indoor localization. The code needs some versioning and cleaning before publishing it as open source on central code repository to be accessed by everyone for further research in the field of indoor localization.

Glossary

Wi-Fi	Wireless Fidelity
RFID	Radiofrequency Identification
RSSI	Received Signal Strength Indicator
UWB	Ultra-wideband
IoT	Internet of Things
WSNs	Wireless Sensor Networks
DBL	Device Based Localization
BLE	Bluetooth Low Energy
EMI	Electromagnetic Interference
RF	Radio Frequency
WSNs	Wireless Sensor Networks
RS-232	Recommended Standard 232
SIG	Special Interests Groups
QoS	Quality of Service
L2CAP	Logic Link Control Adaptation Protocol
RFCOMM	Radio-frequency Communication (Serial Port Emulation)
HCI	Host Controller Interface
SDP	Service Discovery Protocol
GFSK	Gaussian Frequency Shift Keying
LE	Bluetooth Low Energy Audio
ISOC	Isochronous Channels
LEPC	LE Power Control
EATT	Enhanced Attribute Protocol
API	Application Programming Interface
GAP	Generic Access Profile
GATT	Generic Attribute Profile
LUT	Look-Up Table
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
UUID	Universally Unique Identifier
BPSK	Binary Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation
CSMA/CA	Carrier-sense Multiple Access with Collision Avoidance
RC4	Rivest Cipher 4
AP(s)	Access Points
PCMCIA	Personal Computer Memory Card International Association
ISP	Internet Service Providers
MAC	Media Access Control Address
dB	Decibels
ANNs	Artificial Neural Networks
KNNs	K-Nearest Neighbors
CSI	Channel State Information
SLAM	Simultaneous Localization and Mapping
SNR	Signal to Noise Ratio
SOA	State of Art
NN	Neural Networks
SVM	Support Vector Machines
CNN	Convolutional Neural Networks
RF	Random Forests

DTW	Dynamic Time Warping
GUID	Globally Unique Identifier
LTE	Long Term Evolution
Auto-ML	Automated Machine Learning
HTTPS	Hypertext Transfer Protocol Secure
AMQP	Advanced Message Queueing Protocol
MQTT	Message Queueing Telemetry Transport
TTL	Time To Live
SQL	Structured Query Language
AUC	Area Under ROC Curve
ROC	Receiver Operating Characteristics
TP/TN/FP/FN	True Positive/True Negative/False Positive/False Negative
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSLE	Root Mean Squared Logarithmic Error
MAPE	Mean Absolute Percentage Error
R2	R-Squared
SHAP	Shapley Additive Explanations